

First Chinese Forth

WARE TOOLS FOR ADVANCED PROGRAMMERS

Dr. Dobb's Journal

#92 JUNE 1984

\$2.95 [3.50 Canada]



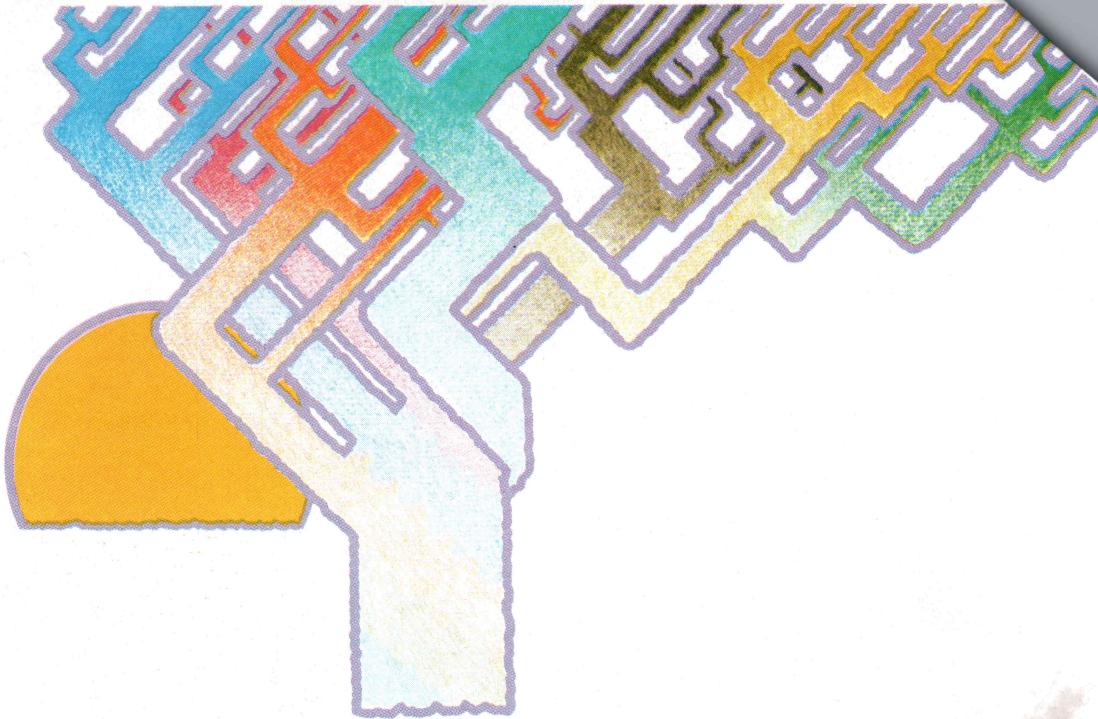
CC
**A Driver
for Small C**

**CP/M on the
Commodore 64**

**Turbo Pascal:
A Review**



06
3835116562



Btrieve.TM

Network and single-user file management for all your programs. All your programming languages.

Fast, flexible b-tree record retrieval.

Say goodbye to writing file management routines.

Because now BtrieveTM can handle file management for all your program development on the IBM PC. All your programming languages. And perform the task better than if you had written your own file management program. With Btrieve, you can develop *better* application programs *faster*.

Based on the b-tree file indexing system, Btrieve provides the most sophisticated file handling powers available for the IBM PC. And for multiple PCs. BtrieveTM/N lets you share files among PCs in MultiLinkTM, PCnetTM, NetWareTM, or EtherSeriesTM local area networks. And both Btrieve and Btrieve/N offer the same superior performance characteristics:

- Interfaces all major IBM PC languages—BASIC, Pascal, COBOL, C, and IBM Macro Assembler
- Written in 8088 Assembler for the IBM PC
- Unlimited number of records per file
- Automatic file recovery on system crash
- User-defined transaction management
- Complete error control and recovery within an application
- Multikey access to records
- Duplicate, modifiable, and segmented keys
- Variable cache buffer from 16K bytes to 64K bytes
- Automatic file organization and b-tree balancing.

Moreover, record retrieval is fast with Btrieve—no matter how large your data base. And with Btrieve's concise, clear documentation, you can begin using Btrieve almost from the moment you load it. So you can begin writing programs *faster* with Btrieve.

Say goodbye to file management routines. And hello to Btrieve.

SC **SoftCraft Inc.**

P. O. Box 9802 #590 Austin, Texas 78766 (512) 346-8380

Suggested retail prices: Btrieve, \$245; Btrieve/N, \$595. Requires PC/DOS or MSTM/DOS, version 1 or 2. Dealer inquiries welcome.

IBM, MS, Btrieve and Btrieve/N, PCnet, MultiLink, NetWare, and EtherSeries are trademarks of International Business Machines, Microsoft Corporation, SoftCraft Inc., Orchid Technology, Davong Systems Inc., Novell Data Systems, and 3Com Corp., respectively.

Please send me information on Btrieve Btrieve/N

Name _____

Company _____

Address _____

City, state, zip _____

Phone _____

Send to SoftCraft Inc., P. O. Box 9802 #590, Austin, TX 78766

Turn
the page.

We're gonna
do a number
on you.

DBASE

III

TM



Circle no. 3 on reader service card.

Introducing the III and only.

It's fast.

It's easy.

It's the most powerful database management system for your 16-bit PC.

And it can do more things with more records in less time.

You've never seen anything like it.

dBASE III™ can handle over a billion records per file, limited only by your computer system. You can have up to ten files open, for sophisticated applications programs.

When you have two related files, information in one can be accessed based upon data in the other.

dBASE III now handles procedures, parameter passing and automatic variables. You can include up to 32 procedures in a single file. With lightning speed. Because once a file is opened, it stays open. And procedures are accessed directly.

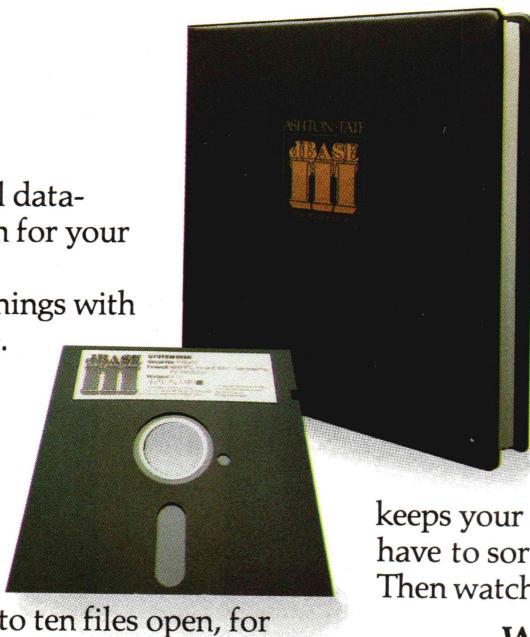
Easier than ever.

dBASE III uses powerful yet simple commands that are the next best thing to speaking English.

If you're unsure of a command, HELP will tell you what to ask for.

If you don't know what command comes next, a command assistant does. All you have to know is what you want it to do.

Our new tutorial/manual will have you entering and viewing data in minutes rather than reading for hours.



And to make matters easier, you get full screen report set-up for simple information access.

Faster than no time at all.

dBASE III isn't just fast.

It's ultra-fast. Operating.

And sorting. Even faster,

is no sorting. Because dBASE III

keeps your records in order so you really don't have to sort anything. Unless you want to. Then watch out!

What about dBASE II®?

It's still the world's best database management system for 8-bit computers. And it's still the industry standard for accounting, educational, scientific, financial, business and personal applications.

Let us do a number on you.

For the name of your nearest authorized dBASE III dealer, contact Ashton-Tate, 10150 West Jefferson Boulevard, Culver City, CA 90230. (800) 437-4329, ext. 333. In Colorado, (303) 799-4900.

ASHTON-TATE



© Ashton-Tate 1984. All rights reserved.

dBASE III is a trademark and dBASE II is a registered trademark of Ashton-Tate.

In This Issue

Dr. Dobb's Journal

Editor-in Chief: *Michael Swaine*

Editor: *Reynold Wiggins*

Managing Editor: *Randy Sutherland*

Contributing Editors:

Robert Blum, Dave Cortesi, Ray Duncan,

Anthony Skjellum, Michael Wiesenber

Copy Editors: *Polly Koch, Cindy Martin*

Typesetter: *Jean Aring*

Circulation and Promotions Director:

Beatrice Blatteis

Fulfillment Manager: *Stephanie Barber*

Direct Response Coordinator: *Maureen Snee*

Promotions Coordinator: *Jane Sharninghouse*

Single Copy Sales Coordinator: *Sally Brenton*

Single Copy Sales: *Lorraine McLaughlin*

Circulation Assistant: *Kathleen Boyd*

Advertising Director: *Susan Cohen Strange*

Advertising Sales:

Alice Hinton, Walter Andrzejewski

Design Director: *Fred Fehlau*

Production/Art Director: *Shelley Rae Doeden*

Production Manager: *Detta Penna*

Production: *Alida Hinton*

Cover Illustration: *Janaia Donaldson*

M&T Publishing, Inc.

Publisher and Chairman of the Board:

Otmar Weber

Director: *C. F. von Quadt*

President: *Laird Foshay*

Entire contents copyright © 1984 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.

Dr. Dobb's Journal (USPS 307690) is published monthly by M&T Publishing, Inc., 2464 Embarcadero Way, Palo Alto, CA 94303, (415) 424-0600. Second class postage paid at Palo Alto and at additional entry points. Address correction requested. Postmaster: Send Form 3579 to Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303. ISSN 0278-6508

Subscription Rates: \$25 per year within the United States, \$44 for first class to Canada and Mexico, \$62 for airmail to other countries. Payment must be in U.S. Dollars, drawn on a U.S. Bank.

Contributing Subscribers: Christine Bell, W. D. Rausch, DeWitt S. Brown, Burks A. Smith, Robert C. Luckey, Transdata Corp., Mark Ketter, Friden Mailing Equipment, Frank Lawyer, Rodney Black, Kenneth Drexler, Real Paquin, Ed Malin, John Saylor Jr., Ted A. Reuss III, InfoWorld, Stan Veit, Western Material Control, S. P. Kennedy, John Hatch, Richard Jorgensen, John Boak, Bill Spees, R. B. Sutton. Lifetime Subscribers: Michael S. Zick, F. Kirk.

Foreign Distributors: ASCII Publishing, Inc. — Japan, Computer Services — Australia, Computer Store — New Zealand, Computercollectief — Nederland, Homecomputer Vertriebs GMBH — West Germany, International Presse — West Germany, La Nacelle Bookstore — France, McGill's News Agency PTY LTD — Australia, Progresco — France.

People's Computer Company

Dr. Dobb's Journal is published by M&T Publishing, Inc. under license from People's Computer Company, Box E, Menlo Park, CA 94026, a non-profit, educational corporation.

Over the next few months you will be seeing some cosmetic changes in *Dr. Dobb's*. Some will be intended to enhance newsstand visibility, some to enhance general readability, some to increase contact or involvement with the readership. You may have noticed when you picked up this issue that the logo is slightly different, as is the descriptor line above the title. In future issues you will see some variation in format for various sections of the magazine, and our conversion in July to a new typesetting system will result in greater readability as well. This issue we have added the section you are presently reading.

With the Editorial page being exclusively devoted to essays each month, we found that we needed some space in which to let you know about things of importance regarding the magazine. This is that space. You will find highlights of the current issue and coming topics, announcements of events or other items of interest, acknowledgements of special contributions to the magazine, or other things we consider noteworthy.

Since we are still settling in, we will postpone details on our growing number of projects, including the bulletin board to which we have alluded, until they progress a bit further. One project that is in full swing, however, is our Fifth Generation Programming Competition. If you did not notice our full-page announcement last month, take a look at page 81 this month for details. In addition, our efforts to improve services should be evident from the growth of the masthead. Many of the new names you will find there are folks who will be making sure that *DDJ* is more available on the newsstands and that subscriptions are serviced better.

The contents page (facing) should give you a good overview of what we have for you this month. During the next few months you can look forward to continued coverage of C and Small C, including a preprocessor for the Small C compiler. Forth will remain an active topic, with our annual Forth issue scheduled for September. Some other items to watch for include an infinite key cryptography system, discussion of Resident System Extensions under CP/M Plus, and some articles on mathematical computation.

Reynold Wiggins

This Month's Referees

Dr. Dobb's Journal regularly draws on the expertise of a Board of Referees for technical evaluation of material submitted for publication. In addition to their remarks to the editors concerning accuracy and relevance of manuscripts, the referees often provide constructive comments for authors regarding clarity or completeness.

The board includes nearly fifty experts from diverse areas of the computer industry and the academic community. Because of space considerations, we can print a list of the entire board only a few times each year. Monthly, however, we do print the names of the referees who contributed their insights on material in that particular issue. Your humble editors, who bear the burden of choosing how material ultimately appears, are grateful for the beneficial insights we receive.

The referees who contributed to this month's issue are:

David E. Cortesi, Contributing Editor, *DDJ*

Kim Harris, Forthright Enterprises

J. E. Hendrix, Office of Computing & Information Services, University of Mississippi

William Ragsdale, President, Dorado Systems

CONTENTS

ARTICLES

14 CP/M on the Commodore 64 — Including Two BIOS Modifications*by Walter G. Piotrowski*

This article presents a brief overview of CP/M on the Commodore 64 along with details of how to patch the BIOS to accommodate two 1541 disk units and/or an RS-232 printer. (Reader Ballot No. 192)

28 dBase II Programming Techniques*by Gene Head*

Departing from last month's discussion of dBase quirks, the author presents a machine-language sub-routine designed for mailing list applications. The technique allows one to partially verify Zip codes and validate states. (Reader Ballot No. 193)

32 First Chinese Forth: A Double-Headed Approach*by Timothy Huang*

Challenged by the task of providing a computer language which could be used by Chinese-speaking people, this author chose Forth as the most logical computer language to implement in Chinese. (Reader Ballot No. 194)

40 cc — A Driver for a Small-C Programming System*by Axel T. Schreiner*

Patterned after the Unix *cc* command, this utility accepts options and file specifications, then uses CP/M's SUBMIT facility to arrange the proper amount of preprocessing, compiling assembly, and loading. (Reader Ballot No. 195)

56 A New Library For Small C (Part II)*by James Hendrix and Ernest Payne*

The listing of the new library for Small C v.2, begun last month, concludes this issue with the second half of the user functions. (Reader Ballot No. 196)

70 Comments on Sixth Generation Computers*by Michael J. Doherty*

The dialogue continues: Doherty responds to Grigonis' essay last month on sixth generation computers. (Reader Ballot No. 197)

DEPARTMENTS

9 Editorial**10 Letters****12 Dr. Dobb's Clinic**

Quasi-Disk; Fooling RMAC; DIR Not-Quite-So-Full; Without Fear or Good Taste (Reader Ballot No. 191)

74 Software Reviews

Turbo Pascal

82 16-Bit Software Toolbox

Professional Basic; The Filepath Utility; Beating the Laws of Thermodynamics; Floating-Point Benchmarks; More on the Microsoft 8086 Assembler; A Toolbox Installment (word_to_hex and byte_to_hex) (Reader Ballot No. 198)

86 C/Unix Programmer's Notebook

8086 memory models; Long pointer package for C compilers (Reader Ballot No. 199)

108 Of Interest

(Reader Ballot No. 200)

110 Advertiser Index



AZTEC C86

Optimized "C" compiler for PC DOS, MS DOS & CP/M-86
 PC DOS, UNIX I/O, math, screen, graphics libraries
 8086 assembler, linker & librarian, overlays
 /PRO—library source, debug, ROM, MASM & RMAC, 8087, large model



NEW C COMPILERS

AZTEC C68K for MACINTOSH
 VAX cross compilers



AZTEC C II

Optimized "C" compiler for CP/M, TRSDOS & LDOS
 assembler, linker & librarian, overlays, utilities
 UNIX I/O, math & compact libraries
 /PRO—library source, ROM, M80 & RMAC

C TOOLS & AIDS

Z editor (like Vi), C TUTOR compiler, PHACT database,
 C GRAFX, UNI-TOOLS I, QUICK C, BABY BLUE for PC
 to CP/M cross, QUADLINK for PC to APPLE cross



AZTEC C65

"C" compiler for APPLE DOS 3.3, ProDOS or COMMODORE 64
 VED editor, SHELL, UNIX & math libraries
 /PRO—library source, ROM, overlays



CROSS COMPILERS

Compile & link on HOST—test on TARGET machine
 HOSTS: UNIX, PC DOS, CP/M-86, CP/M-80, VENIX, PCIX, APPLE
 TARGETS: PC DOS, CP/M-86, CP/M-80, APPLE, RADIO SHACK,
 COMMODORE 64, other hosts and targets available



PRICES

AZTEC C86 C COMPILER

PC DOS MSDOS	249
CP/M-86	249
BOTH	399
/PRO EXTENSIONS	249
Z (VI EDITOR)	125
C TUTOR COMPILER	99
PHACT DATABASE	299
C GRAFX	99
SUPERDRAW	299
UNI-TOOLS I	99
QUICK C	125

AZTEC C II C COMPILER

CP/M	199
/PRO EXTENSIONS	150
TRS 80 MODEL 3	149
TRS 80 MODEL 4	199
TRS 80 PRO (3 & 4)	299

AZTEC C CROSS COMPILERS

PDP-11 HOST	2000
PC DOS HOST	750
CP/M-86 HOST	750
CP/M-80 HOST	750
APPLE HOST	750

AZTEC C65 C COMPILER

APPLE DOS 3.3	199
ProDOS	199
BOTH	299
/PRO EXTENSIONS	99
C TUTOR COMPILER	99
E EDITOR	99
QUICK C	125



MANX SOFTWARE SYSTEMS
 Box 55
 Shrewsbury, NJ 07701
 TELEX: 4995812

TO ORDER OR FOR INFORMATION:
CALL: 800-221-0440 (outside NJ)
201-780-4004 (NJ)

Australia: Blue Sky Industries — 2A Blakesley St. — Chatswood NSW 2067 — Australia 61-2419-5579
 England: TAMSYS LTD — Pilgrim House — 2-6 William St. — Windsor, Berkshire SL4 1BA — England — Telephone Windsor 56747
 Shipping: per compiler next day USA \$20, 2 days USA \$6, 2 days worldwide \$75, Canada \$10, airmail outside USA & Canada \$20
 UNIX is a trademark of Bell Labs. CP/M, CP/M-80 and CP/M-86 are trademarks of DRI. PC DOS is a trademark of IBM. MS DOS is a trademark of MICROSOFT.
 N.J. residents add 6% sales tax.

EDITORIAL

About those dragons.

Dr. Dobb's Journal prides itself on providing useful tools and information for advanced programmers. Last month we ranged beyond the fields we know and published an article on "sixth generation computers," whose author explored the implications of applying bizarre twists of quantum physics to computer design. The discussion of faster-than-light particles with imaginary rest mass will probably never help anyone write a better spreadsheet program or word processor; it frankly bordered on fantasy. Here's why we published it.

China story, version 1.0: China, until the twentieth century, was largely isolated from Western culture. Homage to traditions had kept some strange practices (acupuncture, footbinding) in force while modern science developed in the West. When, in the twentieth century a new generation of Chinese, unwilling to suffer longer the mindbinding of tradition, brought in Western science, the intellectual conquest was swift. Science punctured the fragile structure of ancient traditions, and China joined the twentieth century. Then in 1954 the communists overturned all these advances, exiled the intellectuals to farms and brought back superstition and ignorance as state policy.

China story, version 2.0: China, the oldest civilization on the planet, had a rich structure of tradition. In the twentieth century, overawed by Western technological advances, young Chinese revisionists built a power structure of intellectuals modeled along alien Western lines, and trampled traditional Chinese values. In 1954, the government took action to redress the balance and protect the country's traditions from Western erosion.

Which China story is the truth? Probably neither, but both have been purveyed as truth, and the disparity between the two versions underscores the danger of complacency about one's worldview. Western science is itself a tradition, with its blind spots and unquestioned dogma. It isn't always the best tradition. It is now apparent that some traditional Chinese medical practices work better than their Western counterparts. Unfortunately the blinders of the Western scientific tradition insured that only those scientists who were willing to look into something they *knew* couldn't work were in a position to learn this fact. Those who entertained notions that bordered on fantasy are the ones who learned something new.

It's that way with programming, too.

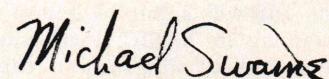
DDJ isn't interested in *just* helping programmers write better spreadsheets and word processors (although we do that.) Programming at its best is a creative activity, and requires that willingness to entertain ideas that can't possibly work. Programming at its best is what *DDJ* is about. So we'll range beyond the fields we know, and sometimes we may trip. This time, apparently, we didn't. As of this writing, our mail is running overwhelmingly in favor of the sixth generation article.

This month we set forth to China, and embark on a different kind of fantasy, with Timothy Huang's article on Chinese Forth. *DDJ* has always championed the cause of putting computer power into the hands of the largest number of people, and placing Forth in the language of a large fraction of Earth's inhabitants is surely a step in that direction. It is of course a fantasy to think that the efforts described in this article will have any significant effect on U.S.-Chinese relations or on our survival on the planet. Isn't it?

Oh yes, the dragons.

This month's cover depicts two dragons, one Western, one Chinese, sharing technology. (The Western dragon is the fat one.) It nicely illustrates our Chinese Forth article. But the dragon is *DDJ*'s symbol, too; one the magazine has, since its first issue, used as a reminder not to be tied to the fields we know. Personal computers are changing the world, and only those who are willing to poke the dragon will find the jewels clinging to its belly.

DDJ has been tickling the dragon for eight years, and sharing the jewels. We think we'll keep right on doing it.



Michael Swaine

LETTERS

The editorial response card is a great way to talk to us, but don't forget that Dr. Dobb's Journal also welcomes letters to the editor as a forum for ideas, innovations, irascibility and even idiosyncrasies. Some letters may be edited for clarity and brevity. The Doctor likes hearing from you - keep on writing.

On Soul of CP/M

Dear DDJ:

I get the feeling that the review of *Soul of CP/M*, Waite & Lafore, was from the viewpoint of an experienced assembly language programmer looking backward. Granted, the number of editorial errors was attackable, but the manner in which the material was presented was tremendous for the target readership.

I've read (or started to read) a good number of CP/M assembly language books. Nealy all of them take the approach of presenting the BDOS calls at the front, numerous multi-page program samples with unintelligible labels and explanations, some cryptic references to the perfect BIOS, and an appendix with the 8080/Z80 instruction set.

Waite & Lafore feed the material in chewable bytes (pun intended) so the reader doesn't need a brain with a Z80 and 64K of permanent memory to comprehend and retain it. No, it is certainly not a complete text, but at least now all those other books make sense to me.

If they can eliminate the remaining editorial errors in the second printing, they have a winner.

Doug Hurst
6808 Estrella Avenue
29 Palms, CA 92277

Even Better Preprocessing

Dear Doctor:

Although this journal may not have many clients - excuse me, readers - who do (or will admit to doing) much programming in BASIC, there must be others like myself who for one reason or another are constrained to use this language (but only during working hours, I protest). All BASIC users owe a debt of gratitude to N. C. Shamas for the NBASIC preprocessor presented in your January issue. This tremendously useful routine has mitigated some of the most serious drawbacks of interpreted BASIC.

There are, however, a couple of improvements that users might wish to make.

The first of these is to make the CASE statement accept strings as expressions, a relatively simple change that requires adding or modifying only four or five lines. The second is a major rearrangement of the program so that CALL statements are processed before the CASE-OF structures. The reason for this is that processing of the CASE-OF commands puts a GOTO on the end of the last line of each group of expressions. CALL statements are resolved into GOSUBs.

Consider, then, the situation when a CALL is the last line of a CASE-OF expression group. As the program is written, the GOSUB is never executed because it becomes preceded by the GOTO. There is another related problem that arises from the appending of a GOTO on the last line of a CASE expression group. If the last line of this group is an IF statement, there must be an ELSE clause included if the CASE construction includes an !ELSE DO statement group. Otherwise, when the IF statement is false, the GOTO will not be executed and the !ELSE DO statement group will be executed. With these two modifications made, and the last quirk kept in mind, NBASIC can quickly become an indispensable part of programming in this most widespread of languages.

Both N. C. Shamas in particular and Dr. Dobb's in general deserve great praise for the quality and usefulness of their articles.

Sincerely,
Dreas Nielsen
234 NW 30th St.
Corvallis, OR 97330

Remarks on RSA

Dear Sirs:

Thank you for a most interesting article on the Rivest-Shamir-Adleman (RSA) Public Key Systems by C. E. Burton in the March 1984 issue. This discourse is well written and certainly very needed. I would, however, like to caution the readers against too heavy a reliance on the security aspects of RSA.

Earlier this year Gustavus Simmons, James Davis, and Diane Holdridge of Sandia National Laboratories were able to factor the last of the Mersenne primes ($2^{251}-1$). By using parallel processing of number clusters it is simple to extrapolate that RSA will no longer be a secure system.

Undoubtedly for general use RSA will continue to be valid. However, with some

of these newly developed techniques the only truly secure system must be the old, one-time pad system, in which the volume of ciphertext is insufficient to analyze statistically, and the key changes with a frequency to prevent valid analysis.

A good basic text for your readers so interested is *Cryptography, A Primer* by Alan G. Konheim (John Wiley & Sons, 1981).

Please keep up the good work. I certainly look forward to your publication each month.

Cordially,
James R. Criscione Jr., M.D.
President
Med-Data, 246 Grand Avenue
Kirkwood, Missouri 63122

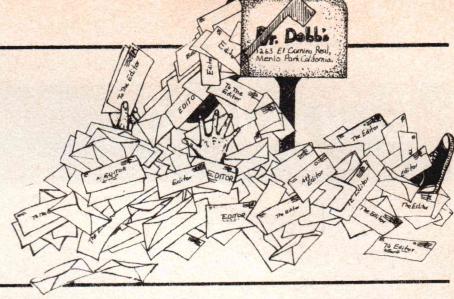
Dear Dr. Dobb:

It was good to hear about an old friend, RATFOR, in Mr. Burton's article on the PKS in the March '84 issue of *Dr. Dobb's*. I have run several tens of thousands of lines of code through the RATFOR preprocessor and was very grateful for its services in an environment where Fortran was the only compiler available. But I am somewhat ambivalent about promoting its usage when there are viable alternatives.

My negative feelings toward RATFOR are summarized by the following factors:

- (1) relatively few people use RATFOR (or have even heard about it) and it is difficult to find a community of users for sharing experiences or software,
- (2) installation of RATFOR can be difficult and occasionally short circuited by versions of Fortran with non-standard limitations,
- (3) the added time for preprocessing on a small micro can be excessive,
- (4) temptations to hack RATFOR into "new and improved" versions can destroy code transportability, and
- (5) we are still using Fortran with all its fundamental limitations.

Fortran has far outlived its useful purpose (if it ever had one) but it is, admittedly, one of the most commonly available compiler languages on minis and midis and maxis. I heartily recommend RATFOR as a salve to soften the agony of having to deal with this unwieldy Tyrannosaurus. Mssrs. Kernighan and Plauger managed to make far better use of their time developing RATFOR than the committee that came up with Fortran '77.



An initial perusal of the article also revealed a typo in the multiplication routine descriptive text: the line "MS" i index should read "i=i-1" not "i=i-1". I also would suggest that Mr. Burton can save some computer time by removing the "mod" and division operations from the multiple-precision add and subtract functions. By using the facts that the sum is always going to be less than twice the radix and the absolute value of the difference is always less than the radix, the C code in Figure (at right) shows "division/mod-less" evaluations.

If we declare:

```
long carry, acc;
unsigned short *u, *v, *w;
```

then we can define RADIX to be "power (2, n)" (2**n for Fortran folks) where n is the number of bits in a type "short" word. This will allow us to compute the equivalent of about 4.5 decimal digits per loop iteration on a machine with 16-bit type "short." Of course, when debugging the system we can define RADIX as 10 to facilitate formatting. I would also tend to make RADIX a compile-time definition rather than a runtime argument for the PKS software.

The last comment relates to Mr. Burton's statement at the end of the addition algorithm text that the Fortran intrinsic "INT(x)" is equivalent to "floor (x)." This is only true for positive values of x. "floor" (related to ALGOL's "entier") is correctly stated as the greatest integer less than or equal to x. Consequently:

floor(-3.45) -> -4

But the Fortran intrinsic yields:

INT(-3.45) -> -3!

I didn't see that Mr. Burton ever used it, but the statement was rather careless and should have been caught in technical review.

Obviously I've found Mr. Burton's article interesting and intend to follow along the steps to producing a public key encryption system as a diversion from some of my more mundane programming activities. It takes me back to the "secret" messages of Capt. Midnight, et al., and the kids' radio shows of the '40s. The trouble is, our decoding rings have gotten a little more expensive. Maybe Dr. Dobb can have "secret" messages about next month's articles.

Sincerely,
Gerald I. Evenden
Box 1027
North Falmouth, MA 02556

```
carry = 0;
while (w >= ws) { /* basic addition loop */
    acc = *u-- + (*v-- + carry);
    *w-- = (carry = acc >= RADIX) ? acc - RADIX : acc;
}

carry = 0;
while (w >= ws) { /* basic subtraction loop */
    acc = *u-- - (*v-- - carry);
    *w-- = (carry = acc < 0) ? acc + RADIX : acc;
}
```

Figure

"Q-PRO 4 blows dBASE II away

*We now complete complex applications
in weeks instead of months. ,*

says Q-PRO4 user, Richard Pedrelli, President, Quantum Systems, Atlanta, GA

" As a dBASE II beta test site the past two years, we were reluctant to even try Q-PRO4. Now we write all our commercial applications in Q-PRO4. We find it to be an order of magnitude more powerful than dBASEII.

We used Q-PRO4's super efficient syntax to complete our Dental Management and Chiropractic Management Systems much faster. Superb error trap and help screen capabilities make our finished software products far more user friendly, too.

In my estimation, any application programmer still using outdated 3rd generation data base managers or worse, a 2nd generation language like BASIC, is ripping himself off. **"**

Runs with PCDOS, MS-DOS, CP/M, MP/M, CP/M86, MP/M86, TurboDOS, MmmOST, MUSE, and NSTAR.

PRICE: 8-bit single-user - \$395; 8-bit multi-user and

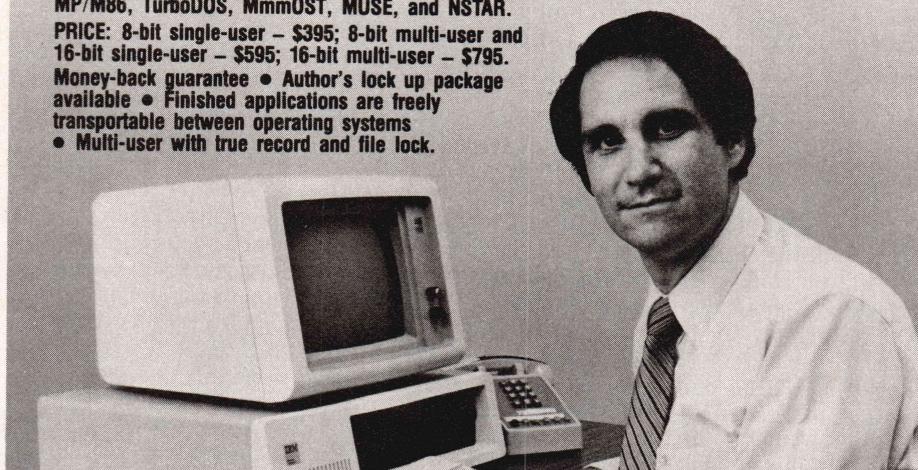
16-bit single-user - \$595; 16-bit multi-user - \$795.

Money-back guarantee • Author's lock up package

available • Finished applications are freely

transportable between operating systems

• Multi-user with true record and file lock.



For Q-PRO 4 demonstration, go to nearest MicroAge store or other fine dealer.

quic-n-easi products inc.

136 Granite Hill Court, Langhorne, PA 19047 (215) 968-5966 Telex 291-765

CP/M, MP/M, CP/M86, and MP/M86 are trademarks of Digital Research. TurboDOS, MmmOST, MUSE, NSTAR, MS-DOS and PC DOS are trademarks of Software 2000, TeleVideo Systems, D.S.M., Molecular, Microsoft and IBM, respectively.

Circle no. 56 on reader service card.

by D.E. Cortesi, Resident Intern

MiniReview: Quasi-Disk

Late last year we decided to reward our S-100 system for three years of faithful work by giving it some new hardware. One thing it had been pining for was an electronic drive. It would be nice to say that we did a careful engineering analysis of the products on the market, but in fact — tell the truth and shame the devil — we merely checked the prices of all the RAM-drive boards we could find in the ads in one month's worth of computer magazines.

The cheapest one was the Quasi-Disk, from Electra-Logics, Inc. (39 Durward Place, Waterloo, Ontario, Canada N2L 4E5; (519) 884-8200). Its retail price of \$799 for 512K was, in December 1983, half that of its nearest competitor. Never ones to let engineering details blind us to the bottom line, we ordered one, plus a battery backup unit at \$159.

The order was held up because of a parts shortage, but Electra-Logics was decent enough to hold our check, only cashing it after it shipped the board early in February. The board arrived in mid-February, and the battery backup unit trailed in in early March.

Once it had arrived, the Quasi-Disk gave a good account of itself. Following the manual, we cut one trace (pin 53), which appears as ground in the IEEE standard but as a disable signal in our system. That done, we slotted the board into the system and fired it up, and the system worked normally. The package included a good diagnostic program that ran correctly under CP/M Plus; it reported that the board was working.

The Quasi-Disk package contains a lot of software (including a self-installing disk driver and a print spooler), but all of it is designed to work with CP/M 2.2. We couldn't test it because we are on CP/M Plus exclusively. It took us several hours to ferret out the details of how to read and write the board as a pseudo-disk and to create a new module for the CP/M Plus BIOS. Installing the code wasn't hard, however, since the CP/M Plus BIOS is modular. All the Quasi-Disk code went into a separate assembly; the names of its public entry points went into the drive table; the BIOS was relinked and a new system generated; and we were up.

We made the board appear to the rest of the system as a disk of 64 tracks, each holding eight 1K physical sectors. The distributed software treats the "disk" as one having 128-byte sectors, but we wanted to reduce the traffic between the BDOS

and the BIOS. Whereas the distributed code uses a 1-byte checksum for each "sector," we reserved 2K of the board to hold true 16-bit CRC codes. No CRC errors have been trapped in six weeks of operation.

The Quasi-Disk is much faster than a mechanical drive, but access to it isn't instantaneous. Data transfer takes just as long as it does to a real disk, but there are no seek or rotational delays. We have a disgustingly disk-bound Sort program. When it sorts a 60K file on disk, it takes 15 minutes 25 seconds. When its work files are on the Quasi-Disk, it runs in 9 minutes 25 seconds, 61% of the mechanical time. The extra six minutes of the disk sort are entirely devoted to head motion and rotational delays — a sobering thought.

Speed is not the Quasi-Disk's only benefit. It turned out to be very nice just to have a third drive. Under CP/M Plus, the PROFILE.SUB that runs automatically at boot time can load that drive with the 100K of standard utilities, then set the drive-search path so that the system searches for commands first on drive M, then on drive A. That lets us omit the standard commands from other disks but still allows us to access them, quickly and silently, at any time.

The battery backup unit comprises a transformer that plugs into the wall and a black box, the size of a carton of cigarettes, that is stuffed full of sealed gel cells. A ribbon cable from the black box snakes into the computer, where it plugs into the Quasi-Disk. When system power is off, the transformer keeps the board alive. If the electric service fails, the gel cells can take over for a couple of hours. A longer outage drains the batteries, losing data.

In our judgment, the Electra-Logics Quasi-Disk is a sound, well-made piece of hardware at a very good price. Its documentation is complete, although not well organized. Its supporting software is entirely oriented to CP/M 2.2, so the CP/M Plus user has to be able to roll his own. That's a pity, because it's the flexibility of CP/M Plus that makes the Quasi-Disk a really convenient extension of the system.

Fooling RMAC

Relocating assemblers such as Digital Research's RMAC make a distinction between relocatable values and nonrelocatable ones. The difference is hard to grasp at first, and if you fail to grasp it,

you may be sorely puzzled by the assembler's actions. Even when you do grasp it, the assembler's rules on what you can and can't do with a relocatable value can cramp your style.

A label is a relocatable value. Define a label:

```
Here: mvi a,1
```

Now the value of the symbol "Here" has been set as a 16-bit number, the contents of the assembler's location counter at that point. The number is a relocatable value, however. The value of "Here" is only the relative offset of "Here" within its particular segment. "Here" won't be at that location when the program runs; its actual storage address will be offset by some amount that can't be known at this time.

Define two labels in the same segment:

```
Here: mvi a,1
```

```
There: ora b
```

Both are relocatable. The assembler will permit you to subtract them,

```
dw There-Here
```

because the difference between them is a constant — it will be the same no matter where the linker puts the program. That is true only when the labels are defined in the same segment, of course. The relative origins of the code and data segments can't be known at assembly time, so the difference between a code label and data label can't be known.

The assembler also can allow you to add or subtract a constant from a relocatable value, so:

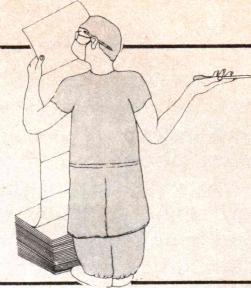
```
dw There-16
```

```
dw Here+5
```

It can permit that because a linker that supports the standard REL-file format must permit adding a constant to a relocatable value. The algebraic addition will be performed at link time; the assembler only needs to record a reference to the (future) location of the label and to record the constant.

No other manipulations are permitted on relocatable values. The value of, say, Here AND OFFh can't be known at assembly time. Nor can the sum of two relocatable values be known. You can usually get around these things, if only by coding instructions that will carry out the evaluation when the program is executed.

What can't be gotten around is the need to evaluate macro operands that are relocatable values. In particular, RMAC doesn't allow any relocatable value to be



evaluated in an IF statement.

This caused us a problem. We were making a set of macros in which we wanted either to save and load a register or leave it alone before doing something. In other words, we wanted to code something along these lines:

```
if ... some test of an operand ...
  push d
  lxi d,operand
endif
```

One way to do this is to allow an omitted operand to mean one thing and a stated operand the other:

```
if not nul operand
  push d
  lxi d,operand
endif
```

But omitting operands is tricksy and makes the code hard to read. It would be a lot nicer, and a lot harder to err, if the user had to write some special keyword — say, “noload” — to specify that the register is already loaded.

Now, the DRI assemblers provide no method by which you can test the literal, superficial encoding of a macro operand. There is no way to code the test, “If the user wrote exactly ‘noload’ then....” Some assemblers for the 8086 do allow that, using the angle brackets as meta-quotes. But in MAC and RMAC, if you code

```
if operand ne noload
```

the assembler will compare the assigned values of “operand” and “noload,” *not* the character strings that name them.

Very well, we thought, we will define our keywords in equate statements, giving them highly unlikely values, e.g.,

```
noload equ 0abcdh
```

Now we can write

```
if operand ne noload
```

and the assembler will compare the value substituted for “operand” to the value equated to “noload,” and we will have our test.

Nope. The value assigned to “noload” is nonrelocatable. If the value substituted for “operand” is relocatable, the IF statement will produce an error message, because we are trying to mix relocatable and nonrelocatable values in one expression.

Oh dear, are we stumped? Not for long. A careful reading of the rules for the substitution of macro operands into macro parameter names reveals an out.

If a macro operand is coded with a leading percent sign, the assembler will evaluate it and substitute not the value with its attached attribute of relocatability, but a simple string of decimal digits. That leads us to the following relocation-stripping macro:

```
unrel macro operand
```

```
@unrel set operand
```

```
endm
```

It does nothing but store the value it was given in a global name. The trick is in how it is used. Within our main macro, where we want to compare for the value of keyword, we now write

```
unrel %operand
if @unrel ne noload
  push d
  lxi d,operand
endif
```

The percent sign strips the attributes from “operand.” Whatever happens to be substituted for it, be it a relocatable label or not, the value assigned to “@unrel” is a simple constant. That can be compared to the equated value of “noload” without error. The full treatment ends up as:

```
some macro operand
```

```
if not nul operand
  unrel %operand
  if @unrel ne noload
    push d
    lxi d,operand
  endif
  else
    +++ operand required as
    address or 'noload'
    exitm
  endif
  ... etc, etc,
  if @unrel ne noload
    pop d
  endif
endifm
```

The user is forced to code some operand. If the operand is omitted, the invalid expression “+++” forces the assembler to display the message line. In that line, the keyword is quoted. If it weren’t, the assembler would substitute for it, producing the error message “operand required as address or 43981,” 43981 being the value 0abcdh equated to “noload.”

DIR Not-Quite-So-Full

CP/M Plus has an extensive DIR command, one that can produce a wide variety of displays. It is very useful, but it is also a trifle verbose. When you ask for a full display, you get:

- a blank line
- a line, “Scanning Directory”
- another blank line
- a line, “Sorting Directory”
- another blank line
- column headings
- another blank line
- a line of dashes
- another blank line

Then, finally, you see the display of files, sizes, timestamps, etc. These extra lines are unnecessary. Often they push off the screen previous commands that one would like to refer back to.

We worked out how to stifle some of these lines. If you (the six of you that have CP/M Plus) would like to stifle them as well, you can do it this way. Load up DIR.COM under SID, using this sequence:

```
rename dir.old=dir.com
sid dir.old
```

Check these five hex addresses for the given instructions:

```
2091 CALL 30E2
24F7 CALL 30E2
2DF0 CALL 272C
2DF6 CALL 272C
2E2D CALL 272C
```

If your DIR.COM is the same as ours, that’s what you’ll find. The first two calls produce the “Scanning” and “Sorting” messages and the blank lines that follow them, while the others produce blank lines around the heading.

Replace each CALL with three NOP instructions, then write a new DIR file, using

```
-wdir.com
```

Test the command. It should produce a more compact display.

Without Fear or Good Taste

Dear Reader, the ol’ mailbag is running dry. What has your system done to surprise you lately? What has your operating system done to disgust you, and how did you get even with it? Know any good puzzles? Gotta have input, gotta get some of the cobwebs off the chairs in the waiting room at the old Clinic.

Hey, I know, let’s have a contest! (Picture Mickey Rooney: “Hey, gang, I know — we’ll put on a show!”) Sure. Let’s hold the first (and last) annual ZOSO sound-alike contest. Surely you remember ZOSO, the vile-tempered person who used to hold forth with outrageous, near-libelous, dead-accurate opinions on the old CPMUG disks?

If you can’t think of anything else to write about (puzzles, system discoveries, perils and pitfalls), then send us your best imitation of ZOSO: a page of scornful, acerbic, sneering commentary on some aspect of the personal-computer scene. We’ll print the best ones that our lawyer will approve. You’ll feel *much* better afterward.

BBJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 191.

CP/M on the Commodore 64

Including Two BIOS Modifications

After the recent outcry about the correct spelling of the word kernel, we thought it interesting that the documentation for the Commodore 64 spelled the word kernal. Because this article refers to the Commodore "kernal" specifically, we have decided to use Commodore's spelling in this instance. — Ed.

The Commodore 64 has the potential to become one of the least expensive "real" computer systems available. Although most people know that the C-64 has a poor version of BASIC and that the OS doesn't really qualify as an operating system, not as many people know that they don't have to be a captive of Commodore's firmware.

The C-64 contains 64K of real RAM. Some of it is hidden under the ROM that holds the BASIC and the ROM that holds the OS kernal, but any program can switch out both of these ROMs to give access to the RAM underneath. Commodore is marketing a CP/M package for the C-64 that takes partial advantage of this hidden RAM by switching out the BASIC but leaving the OS I/O kernal switched in. Their implementation results in a computer that can run a 48K version of CP/M 2.2 (Although it is not the subject of this article, I'm convinced that a larger version of CP/M is possible with a rewrite of the BIOS.)

The CP/M package, which is similar in concept to the CP/M for the Apple, consists of a Z80 co-processor that plugs into the expansion port on the back of the keyboard and a CP/M system diskette. When the C-64 is running CP/M, the Z80 and the C-64's native processor (the 6510) take turns running out of the common memory. The 6510 becomes a slave processor; the Z80 turns it on whenever it requires I/O. Communication between the two processors is through data and commands left in memory. As with all other CP/Ms, only the BIOS is customized for the Commodore; the BDOS is not aware of the existence of the 6510.

The CP/M system is started with the C-64 running in its native mode. After plugging the CP/M cartridge into the back of the C-64, the user loads a CP/M boot program, called CPM, from the system diskette just as if it were a BASIC program. This boot program, called BOOT65 in the Commodore CP/M manual, loads in two other routines. The first is the 6510 portion of the BIOS (BIOS65), and the second is a Z80 boot routine (BOOT80). Once both routines are in place, BOOT65 switches on the Z80, and from this point on the Z80 is in charge. The Z80 boot routine loads in the Z80 portion of the BIOS (BIOS80) and the rest of CP/M.

The interface between the two parts of the BIOS is very nicely designed and very clean. Ten memory locations serve as a communication region, and a 256-byte buffer holds a full C-64 disk sector. When the Z80 part of the BIOS needs some physical I/O, it puts the required command information in the communication area (locations \$900-\$90A in the 6510 address space), switches itself off, and turns the 6510 on. The 6510 picks up the command, does what is required, then switches back to the Z80. The hardware is arranged so that both computers resume execution at the instruction following the instruction that performed the switch (plus one byte).

Because both processors required access to a set of low-numbered memory addresses to function reasonably, the address spaces in the two machines are offset from each other. The 6510 keeps its low memory data in the location adjacent to the "real" zero memory location. As shown in Figure 1 (page 15), the zero memory location for the Z80 is actually at location \$1000 in this 6510 address space. Because of the offset of \$1000, the communication area at \$900 in the 6510 address space is at location \$F900 in the Z80 space (see Figure 2, page 15). In other words, \$1000 is added to every Z80 address; hence, a Z80 access to \$F900 yields \$0900, and a Z80 access to \$0000 reaches \$1000.

Adding a Second Drive

I get the impression that Commodore rushed to market with CP/M. The software and the documentation both have some shortcomings. The biggest problem is that the BIOS can communicate with only one disk unit. It can talk to both drives in a Commodore 4040 unit or to

the single drive in a 1541, but it cannot communicate with two or more 1541 drives. If you have a 1541, you can still use an A and a B "drive" in CP/M, but each time you switch between A and B, you have to swap diskettes in and out of a single 1541. The software in the BIOS always tells you when the swap is required. I have two 1541s, however, and I quickly became annoyed with diskette swapping in one of them while the other sat unused. What I did about it follows.

The manual supplied by Commodore contains complete listings of all of the Z80 code for CP/M, but it does not contain listings for any of the 6510 code. The Z80 listings are well commented and easy to read. Because of this and because of the cleanliness of the interface between the two parts of the BIOS, I was able to follow the 6510 code with the use of a disassembler.

The changes that I made to allow CP/M to use two 1541 drives are relatively simple. They are so simple that I am surprised that Commodore did not include this capability in the original package. These modifications are shown in Listing One (page 18). The first four sections of this listing are patches that replace some of the original BIOS65 code. The last sections are small additional routines that are called by the patches; they fit in a section of memory reserved for the BIOS but actually unused.

The first patch routes the processing of all BIOS65 commands from the Z80 through the new routine called TESTIT. This routine checks to see if the command is a disk operation. If it is, the routine compares the drive number in the command with the drive number that was accessed on the last disk command. If they are the same, control returns to BIOS65. If they are different, the routine just closes the disk channels then returns control to BIOS65. BIOS65 discovers that the channels are closed when it attempts to communicate with the disk and tries to reopen them. The last two patches route the open requests through the new routines called OPEN15 and OPEN2, which open the channels to the correct disk. The changes make minimal impact on the original program. All of the disk errors that are detected and reported back to CP/M still function as they did originally.

A bigger part of the problem was coming up with a way to make the changes permanently on a CP/M diskette.

by Walt Piotrowski

Walt Piotrowski, R. D. 1 Box 582, Afton, NY 13730.

Since my 6510 assembler and loader do not run under CP/M, I chose to make all of the changes on the 6510 in its native mode. This required writing a utility program to read the 6510 code from the diskette and to replace it, once it had been modified, back in the same sectors. This utility turned out to be longer than the changes themselves, although it was easier to write because I didn't have to explore someone else's machine language code first.

The utility (CPM65UT) is provided in Listing Two (page 19). The utility does not have the capability to make changes; it simply reads the 6510 code from the disk or replaces it. Therefore the utility must be used in conjunction with a loader or a machine language monitor. The 6510 code is read into memory by calling the utility at its read entry point (\$C000). It reads BOOT65 and BIOS65 into the locations where they normally reside (see Figures 1 and 2). The loader or monitor is then used to make the modifications. After the changes have been made, the utility is called at its write entry point (\$C003), and it writes the modified code back onto the disk. Its use with the loader in the Commodore Assembler Development System is discussed at the end of this article.

Adding a Serial Printer

My printer is not a Commodore printer. It's a DECwriter that I've had for several years; I modified it for RS-232 operation when I bought the C-64. While I was modifying the BIOS for two-disk operation, I also made a change to incorporate the DECwriter as the system printer. Listing Three (page 26) shows the changes to BIOS65 that make this substitution. The two changes, the one for the disks and the one for the RS-232 printer, are completely independent; you can incorporate either of them separately or use both together.

If you examine the Commodore CP/M manual, you will find that Commodore provides two user-definable functions in the interface between the Z80 and the 6510 parts of the BIOS. They also leave two 256-byte areas in the space reserved for BIOS65 to allow room for the 6510 code that you will write for these functions. Because I wanted to use all of the 48K bytes of memory that are available for CP/M programs and because I had no plans to implement any user functions, I moved the Commodore kernel's RS-232 buffers into one of these areas. Their normal location would have decreased the space available to CP/M by 4K bytes. If you choose to make the changes the same way I have, you will not be able to use the second user function in any program that also uses the printer.

Although it is unlikely, the possibility exists that more than one version of Commodore's CP/M is in circulation. Therefore, I've included comments in the two-disk modification to show the original memory contents for each of the four small patches. If you decide to make the changes, be sure to compare what is in your memory with what the listing

says should be there. If you find a difference and need some help, please let me know and I'll do all I can to assist you. If you decide to make the RS-232 modification, you should note that it is a large patch that completely overlays the printer code in the BIOS. If you find that the disk patch locations match up, you should be able to make the RS-232

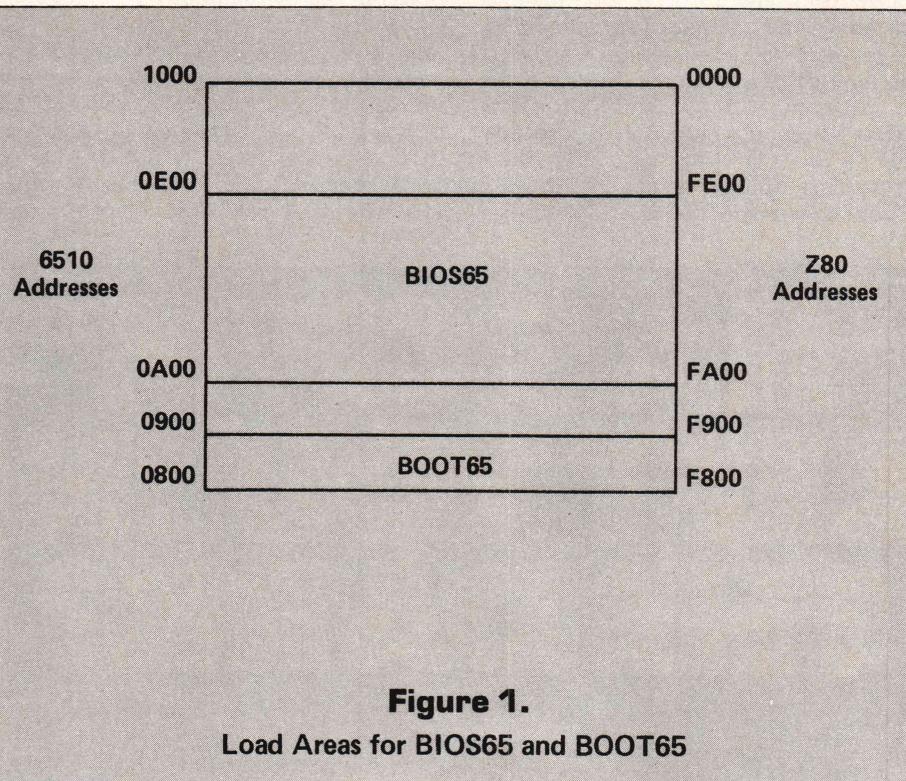


Figure 1.
Load Areas for BIOS65 and BOOT65

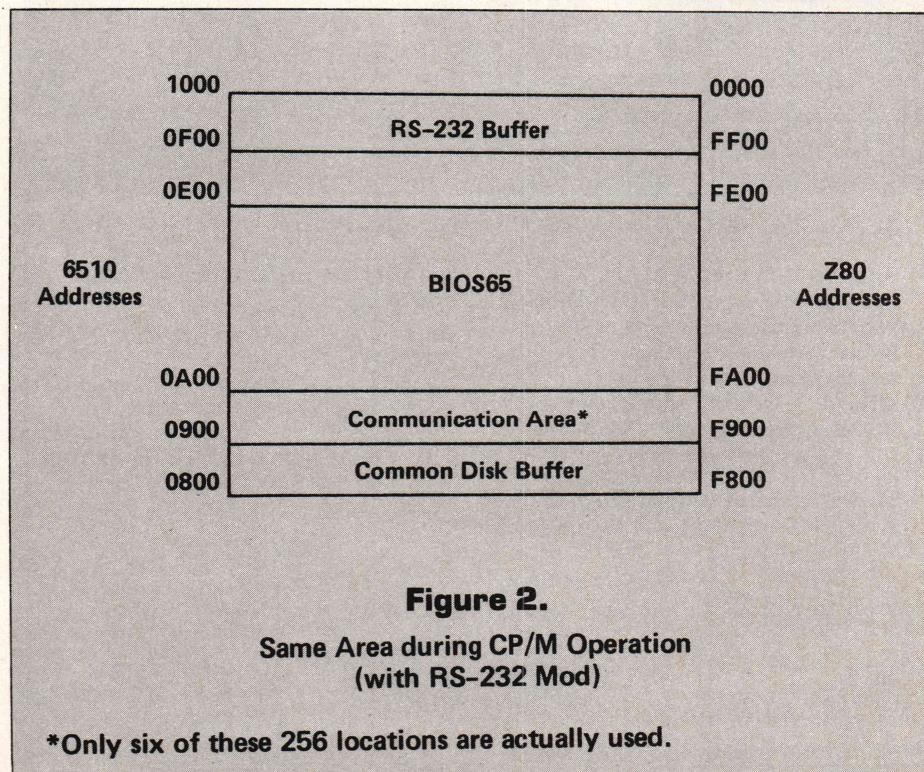


Figure 2.
Same Area during CP/M Operation
(with RS-232 Mod)

*Only six of these 256 locations are actually used.

change easily. If you have a problem, again I will be happy to help out.

After you have made the changes to the BIOS, you must run the CONFIG program that is located on the CP/M diskette. If you have made the disk drive changes, you should change to a two-drive configuration. The modifications made to the BIOS will allow you to change back and forth at any time. If you have CP/M configured for two drives, the program will access drives 8 and 9; if you have CP/M configured for one drive, the program will use drive 8 in the diskette swapping mode. (It is also possible to use more than two 1541 drives by making a minor change in BIOS80.) If you have made the RS-232 change,

you should use CONFIG to change the CP/M configuration so that BIOS80 thinks that you have the Commodore 4022 printer. If you leave it configured for the 1525 printer, BIOS80 makes a conversion from the ASCII character set that is used by CP/M to the 1525's character set.

You should be cautious about two things if you decide to buy Commodore's CP/M. The first is memory size. If you look through the CP/M Software Finder, you will find many programs that will run in 48K, but many also will not. All of the ones that I am now interested in fit in 48K, but this may not be true for you. The second is disk format. Commodore's 1541 disk format is unique, and

the software supplier for the program that you are interested in may not have this format available yet. I believe that this is a temporary problem and that it will be solved as the suppliers begin to realize that there are a lot of C-64 owners in the world.

As a final note, there is also the possibility of downloading public domain software from a bulletin board or another CP/M system using the inexpensive modems available for the C-64. I'm not aware of any software currently available for Commodore's CP/M that will do that, but I would like to hear from you if you know of any. If not, it doesn't look as if it would be that tough a program to write, making use of the user functions that are provided in the interface between BIOS65 and BIOS80. Any volunteers?

Using CPM65UT with the Commodore Assembler Development System

The Commodore Assembler Development System is a software package that contains an editor, an assembler, a machine language monitor, and a loader, along with several other useful programs. The package contains two versions of the loader. The version called HILOADER64 resides in memory at \$C800 (51200 decimal). If you have previously assembled the CP/M changes, you can load both the CPM65 utility and the loader into memory at the same time. You then execute the utility by typing SYS49152. The utility will read the 6510 code into memory. When this is complete, you execute the loader by typing SYS51200. The loader will ask the name of the object file that you want to load. When loading is complete, you write the modified code back onto disk by typing SYS49155.

References

Commodore Business Machines, *Commodore 64 CP/M Operating System User's Guide*, Howard W. Sams and Co., Indianapolis, IN 46268, 1983.

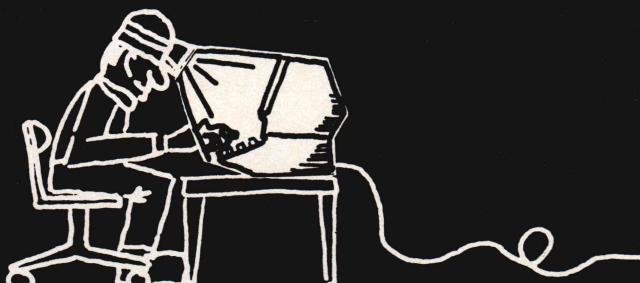
Commodore Business Machines, *Commodore 64 Programmer's Reference Guide*, Howard W. Sams and Co., Indianapolis, IN 46268, 1983.

Digital Research, *CP/M Software Finder*, Que Corporation, Indianapolis, IN 46250, 1983.

Rodney Zaks, *The CP/M Handbook with MP/M*, Sybex Corp., Berkeley, CA 94710, 1980.

BBJ

HAS CP/M® LEFT YOU IN THE DARK?



With the MRS/OS Source Code, you can see the light.

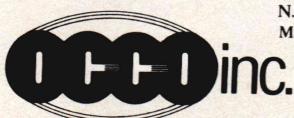
If you own a CP/M compatible operating system, you've had to put up with the mistakes and quirks of someone else's programming. Until now. Now you can see the light with MRS/OS. In fact, MRS is a full operating system designed to replace CP/M 2.2 or CDOS and it comes with complete source code. MRS is designed for Z80 processors, runs CP/M software, and can interface directly to a CP/M BIOS, saving you a lot of sysgen time.

All this for under sixty bucks.

With MRS, you get more than what you pay for. For under sixty dollars you receive fully commented source code for standard and extended BDOS functions, a sample BIOS, our all-in-one utility package and a 150 page manual.

So if you're tired of being in the dark with some other guy's program, here's the answer to your prayers.

\$59.95 complete
(includes shipping & handling in N. America; overseas add \$12)
Mass. orders include 5% sales tax



16 Bowman Lane
Westboro, MA 01581
(617) 366-8969

Orders: 9 am - 10 pm EST
Tech. inquiries: 7 pm - 10 pm EST

CP/M is a registered trademark of Digital Research Corp.

CDOS is a registered trademark of Cromemco Corp.

Circle no. 43 on reader service card.

(Listings begin on page 18)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 192.

Don't call her cheap. Call her beautiful.

The Bonnie BlueTM

Word Processing System for the IBM Personal Computer

It's obvious what makes her so cheap, but what makes Bonnie Blue so beautiful? Bonnie Blue is a new and easy-to-use word processing program for the IBM Personal Computer.

The Full System. The Bonnie Blue System includes in one program a full screen Editor, a Printing module and a useful Toolbox. It includes the features you've come to expect, and more:

complete cursor control: by character, word, line; page up and down instantly; go to top, bottom of document; auto scroll towards top or bottom

word wrap

margin justification, centering

adjustable margins, tabs, indents

reformat paragraphs

move, copy, delete, paste blocks

find with delete, insert, replace and wild card characters

keyboard remapping

multi-line headers, footers

Bonnie Blue can handle lines longer than the screen is wide, by horizontally scrolling the line. And, unlike some programs, Bonnie Blue lets you include any displayable character in your text, such as block graphics and foreign language characters.

Unique Features. With Bonnie Blue, you can "paint" display attributes onto your text, by the character, word, or line, or automatically as you enter text. With the monochrome adapter, you can paint any combination of underlined, bold, reverse video or blinking. With an 80 column monitor and the color/graphics adapter, this translates into a palette of 16 color combinations to choose from. And if your computer has both monitors, Bonnie Blue lets you use them both, shifting back and forth as you wish.

IBM Personal Computer is a trademark of IBM Corp. Epson Graftax Plus is a trademark of Epson America Inc.

Bonnie Blue Software

Post Office Box 536
Liverpool, NY 13088

Send me the Bonnie Blue System. I am enclosing \$50 (NY State residents please add 7% sales tax).

Please send literature.

I have a _____

Check enclosed VISA MasterCard Sorry, no COD.

Credit Card No. _____ Expires _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Company _____

Only \$50

Minimum recommended system:
IBM PC, 128K, 2 disk drives,
PC-DOS 1.1 or 2.0, 80-column
monitor or monochrome adapter,
or both, Epson MX-80 or
MX-100 with Graftax Plus.

*Versions available soon for PCjr.
Write for details.*

CP/M on the Commodore 64 (Text begins on page 14)

Listing One

CPMDISK.TX.

LINE#	LOC	CODE	LINE	
00001	0000		*****	
00002	0000		;	
00003	0000		; MODIFICATIONS TO BIOS65 FOR	
00004	0000		; TWO 1541 DISKS (8 AND 9)	
00005	0000		;	
00006	0000		; W.G. PIOTROWSKI	
00007	0000		; STATE UNIV OF NY	
00008	0000		; BINGHAMTON, NY 13901	
00009	0000		;	
00010	0000		*****	
00011	0000		;	
00012	0000	BIOS65 =\$0A0C	; BIOS RETURN LOC	
00013	0000	COMMAND =\$0900	; COMMAND LOC	
00014	0000	DISKNO =\$0904	; DISK NUM LOC	
00015	0000	CLOSE =\$FFC3	; KERNAL CLOSE ROUTINE	
00016	0000	;		
00017	0000	*=\$0A06	; CHECK FOR DISK OPS	
00018	0A06	20 B0 0C	JSR TESTIT	; WAS JSR \$0A0C
00019	0A09	;		
00020	0A09	*=\$0AFB	; REMOVE DISK NUM STORE	
00021	0AFB	EA	NOP	; WAS STA \$0B67
00022	0AFC	EA	NOP	;
00023	0AFD	EA	NOP	;
00024	0AFE	;		
00025	0AFE	*=\$0B9C	; BIOS TRYING TO OPEN 15	
00026	0B9C	20 90 0C	JSR OPEN15	; WAS LDA #15
00027	0B9F	EA	NOP	;
00028	0BA0	EA	NOP	;
00029	0BA1	EA	NOP	;
00030	0BA2	;		
00031	0BA2	*=\$0BB6	; BIOS TRYING TO OPEN 2	
00032	0BB6	20 A0 0C	JSR OPEN2	; WAS LDA #2
00033	0BB9	EA	NOP	;
00034	0BBA	EA	NOP	;
00035	0BBB	EA	NOP	;
00036	0BBC	;		
00037	0BBC	*=\$C90	; FREE SPACE IN BIOS AREA	
00038	0C90	A9 0F	OPEN15 LDA #15	; COMMAND CHANNEL
00039	0C92	20 C3 FF	JSR CLOSE	; CLOSE-JUST IN CASE
00040	0C95	AD 04 09	LDA DISKNO	; GET DISK NUM
00041	0C98	18	CLC	; SET UP FOR ADD
00042	0C99	69 08	ADC #8	; MAKE 8 OR 9
00043	0C9B	AA	TAX	; DEVICE # IN X
00044	0C9C	A9 0F	LDA #15	; CHANNEL IN A
00045	0C9E	A8	TAY	; ALSO IN Y
00046	0C9F	60	RTS	; GO BACK-CALL SETLFS
00048	0CA0	A9 02	OPEN2 LDA #2	; DATA CHANNEL
00049	0CA2	20 C3 FF	JSR CLOSE	; CLOSE-JUST IN CASE
00050	0CA5	AD 04 09	LDA DISKNO	; GET DISK NUM
00051	0CA8	18	CLC	; SET UP FOR ADD
00052	0CA9	69 08	ADC #8	; MAKE 8 OR 9
00053	0CAB	AA	TAX	; DEVICE # IN X
00054	0CAC	A9 02	LDA #2	; CHANNEL IN A
00055	0CAE	A8	TAY	; ALSO IN Y

00056	0CAF	60	RTS	; GO BACK-CALL SETLFS
00057	OCB0		;	
00058	OCB0		;	
00059	OCB0	AD 00 09	TESTIT LDA COMAND	; SEE IF DISK COMND
00060	OCB3	FO 08	BEQ CLOSIT	; 0 IS DISK CMD
00061	OCB5	C9 01	CMP #1	
00062	OCB7	FO 04	BEQ CLOSIT	; 1 IS DISK CMD
00063	OCB9	C9 06	CMP #6	; 6 IS DISK CMD
00064	OCBB	DO 15	BNE LEAVE	; NOT FOR DISK-EXIT
00065	OCBD	AD 04 09	CLOSIT LDA DISKNO	; GET CMD DISK#
00066	OCCE	CD D5 0C	CMP LSTDSK	; COMPARE WITH LAST USE
00067	OCCE	FO 0D	BEQ LEAVE	; SAME-LET GO
00068	OCCE	BD D5 0C	STA LSTDSK	; DIFF-SAVE FOR NEXT
00069	OCCE	A9 0F	LDA #15	; COMMAND CHANNEL
00070	OCCE	20 C3 FF	JSR CLOSE	; CLOSE IT
00071	OCCE	A9 02	LDA #2	; DATA CHANNEL
00072	OCCE	20 C3 FF	JSR CLOSE	; CLOSE IT
00073	OCB2	4C 0C 0A	LEAVE JMP BIOS65	; BIOS WILL OPEN AGAIN
00074	OCB5		;	
00075	OCB5	00	LSTDSK .BYTE 0	
00076	OCB6		.END	

ERRORS = 00000

SYMBOL TABLE

SYMBOL VALUE

BIOS65	0A0C	CLOSE	FFC3	CLOSIT	OCBD	COMAND	0900
DISKNO	0904	LEAVE	OCB2	LSTDSK	OCB5	OPEN15	0C90
OPEN2	0CA0	TESTIT	OCB0				

END OF ASSEMBLY

End Listing One

Listing Two

CPM65UT.TX.

LINE#	LOC	CODE	LINE
00001	0000		*****
00002	0000		;
00003	0000		; UTILITY PROGRAM TO READ OR WRITE
00004	0000		; THE 6510 PORTION OF THE CPM BIOS
00005	0000		; AND THE 6510 CPM BOOT ROUTINE
00006	0000		;
00007	0000		W. PIOTROWSKI
00008	0000		;
00009	0000		; TO READ - JSR \$C000 SYS(49152)
00010	0000		; TO WRITE - JSR \$C003 SYS(49155)
00011	0000		; DISK DRIVE # AT \$C006 (49158)
00012	0000		;
00013	0000		*****
00014	0000		;
00015	0000		; EQUATES
00016	0000		;
00017	0000	CMDCHN =15	; DISK COMMAND CHANNEL
00018	0000	DATCHN =2	; DISK DATA CHANNEL

(Continued on next page)

CP/M on the Commodore 64

(Listing Continued, text begins on page 14)

Listing Two

00019	0000	ZERO	=\$30	;ASCII ZERO
00020	0000	ONE	=\$31	;ASCII ONE
00021	0000	TWO	=\$32	;ASCII TWO
00022	0000	CR	=\$D	;ASCII RETURN
00023	0000		;	
00024	0000	SETLFS	=\$FFBA	;KERNEL ROUTINES
00025	0000	SETNAM	=\$FFBD	;
00026	0000	OPEN	=\$FFC0	;
00027	0000	CLOSE	=\$FFC3	;
00028	0000	CHKIN	=\$FFC6	;
00029	0000	CHKOUT	=\$FFC9	;
00030	0000	CLRCHN	=\$FFCC	;
00031	0000	CHRIN	=\$FFCF	;
00032	0000	CHROUT	=\$FFD2	;
00033	0000		;	
00034	0000	FREKZ1	=\$FB	;FREE SPACE PAGE ZERO
00035	0000	FREKZ2	=FREKZ1+2	;4 LOCS AVAIL
00036	0000	BOOT65	=\$801	;NORMAL BOOT65 START
00037	0000	BUF	=\$900	;INPUT BUFFER START
00038	0000	BTSTRT	=\$904	;BOOT65 START IN BUF
00039	0000	BIOS65	=\$A00	;BIOS65 LOAD ADDRESS
00040	0000	BOOTLN	=BIOS65-BTSTRT	;LENGTH OF BOOT65
00041	0000		;	
00042	0000		;	MACRO DEF
00043	0000		;	
00044	0000		.	MAC XAD
00045	0000		LDX	#\$<?1
00046	0000		LDY	#\$>?1
00047	0000		STX	?2
00048	0000		STY	?2+1
00049	0000		.	MND
00051	0000		*	=\$C000
00052	C000		;	
00053	C000		;	ENTRY POINTS
00054	C000		;	
00055	C000	4C 07 C0	JMP READ	;(\$49152)
00056	C003	4C OF C0	JMP WRITE	;(\$49155)
00057	C006	09	DISK	.*BYTE 9
00058	C007		;	DISK DRIVE *
00059	C007	A9 00	READ	LDA #0
00060	C009	8D 57 C1		STA INOUT
00061	C00C	4C 14 C0		JMP START
00062	C00F	A9 01	WRITE	LDA #1
00063	C011	8D 57 C1		STA INOUT
00064	C014		;	
00065	C014		;	OPEN DISK CHANNELS
00066	C014		;	
00067	C014	78	START	SEI
00068	C015	A9 OF		LDA #CMDCHN
00069	C017	AE 06 C0		LDX DISK
00070	C01A	A8		TAY
00071	C01B	20 BA FF		JSR SETLFS
00072	C01E	A9 00		LDA #0
00073	C020	20 BD FF		JSR SETNAM
00074	C023	20 C0 FF		JSR OPEN
00075	C026		;	
00076	C026	A9 02		LDA #DATCHN
				DATA CHANNEL

(Continued on page 22)

Z80 Software

SOFTWARE DESCRIPTIONS

TPM (TPM I) - \$80 A Z80 only operating system which is capable of running CP/M programs. Includes many features not found in CP/M such as independent disk directory partitioning for up to 255 user partitions, space, time and version commands, date and time, create FCB, chain program, direct disk I/O, abbreviated commands and more! Available for North Star (either single or double density), TRS-80 Model I (offset 4200H) or II, Versafloppy I, or Tarbell I.

TPM-II - \$125 An expanded version of TPM which is fully CP/M 2.2 compatible but still retains the extra features our customers have come to depend on. This version is super FAST. Extended density capability allows over 600K per side on an 8" disk. Available preconfigured for Versafloppy II (8" or 5"), Epson QX-10, Osborne II or TRS-80 Model II.

CONFIGURATOR I

This package provides all the necessary programs for customizing TPM for a floppy controller which we do not support. We suggest ordering this on single density (8SD).

Includes: TPM-II (\$125), Sample BIOS (BIOS) SOURCE (\$FREE), MACRO II (\$100), LINKER (\$80), DEBUG I (\$80), QED (\$150), ZEDIT (\$50), TOP I (\$80), BASIC I (\$50) and BASIC II (\$100)

\$815 Value

NOW \$250

CONFIGURATOR II

Includes: TPM-II (\$125), Sample BIOS (BIOS) SOURCE (\$FREE), MACRO II (\$100), MACRO III (\$150), LINKER (\$80), DEBUG I (\$80), DEBUG II (\$100), QD (\$200), QED (\$150), ZTEL (\$80), TOP II (\$100), BUSINESS BASIC (\$200) and MODEM SOURCE (\$40) and DISASSEMBLER (\$80)

\$1485 Value

NOW \$400

MODEL I PROGRAMMER

This package is only for the TRS-80 Model I. Note: These are the ONLY CDL programs available for the Model I. It includes: TPM I (\$80), BUSINESS BASIC (\$200), MACRO I (\$80), DEBUG I (\$80), ZDDT (\$40), ZTEL (\$80), TOP I (\$80) and MODEM (\$40)

\$680 Value

NOW \$175

MODEL II PROGRAMMER

This package is only for the TRS-80 Model II. It includes: TPM-II (\$125), BUSINESS BASIC (\$200), MACRO II (\$100), MACRO III (\$150), LINKER (\$80), DEBUG I (\$80), DEBUG II (\$100), QED (\$150), ZTEL (\$80), TOP II (\$100), ZDDT (\$40), ZAPPE SOURCE (\$80), MODEM (\$40), MODEM SOURCE (\$40) and DISASSEMBLER (\$80)

\$1445 Value

NOW \$375

BASIC I - \$50, a 12K+ basic interpreter with 7 digit precision.

BASIC II - \$100. A 12 digit precision version of Basic I.

BUSINESS BASIC - \$200. A full disk extended basic with random or sequential disk file handling and 12 digit precision (even for TRIG functions). Also includes PRIVACY command to protect source code, fixed and variable record lengths, simultaneous access to multiple disk files, global editing, and more!

ACCOUNTING PACKAGE - \$300. Written in Business Basic. Includes General Ledger, Accounts Receivable/Payable, and Payroll. Set up for Hazeline 1500 terminal. Minor modifications needed for other terminals. Provided in unprotected source form.

MACRO I - \$80. A Z80/8080 assembler which uses CDL/TDL mnemonics. Handles MACROs and generates relocatable code. Includes 14 conditionals, 16 listing controls, 54 pseudo-ops, 11 arithmetic/logical ops, local and global symbols, linkable module generation, and more!

MACRO II - \$100. An improved version of Macro I with expanded linking capabilities and more listing options. Also internal code has been greatly improved for faster more reliable operation.

MACRO III - \$150. An enhanced version of Macro II. Internal buffers have been increased to achieve a significant improvement in speed of assembly. Additional features include line numbers, cross reference, compressed PRN files, form feeds, page parity, additional pseudo-ops, internal setting of time and date, and expanded assembly-time data entry.

DEVELOPER I

Includes: MACRO I (\$80), DEBUG I (\$80), ZEDIT (\$50), TOP I (\$80), BASIC I (\$50) and BASIC II (\$100)

\$440 Value

NOW \$150

DEVELOPER II

Includes: MACRO II (\$100), MACRO III (\$150), LINKER (\$80), DEBUG I (\$80), DEBUG II (\$100), BUSINESS BASIC (\$200), QED (\$150), TOP II (\$100), ZDDT (\$40), ZAPPE SOURCE (\$80), MODEM SOURCE (\$40), ZTEL (\$80), and DISASSEMBLER (\$80).

\$1280 Value

NOW \$350

DEVELOPER III

Includes: QD (\$200), QED (\$150), BUSINESS BASIC (\$200), ZTEL (\$80) and TOP II (\$100)

\$730 Value

NOW \$300

COMBO

Includes: DEVELOPER II (\$1280), ACCOUNTING PACKAGE (\$300), QD (\$200) and 6502X (\$150)

\$1930 Value

NOW \$500

LINKER - \$80. A linking loader for handling the linkable modules created by the above assemblers.

DEBUG I - \$80. A tool for debugging Z80 or 8080 code. Disassembles to CDL/TDL mnemonics compatible with above assemblers. Traces code even through ROM. Commands include Calculate, Display, Examine, Fill, Goto, List, Mode, Open File, Put, Put Wait, Trace, and Search.

DEBUG II - \$100. A superset of Debug I. Adds Instruction Interpreter, Radix change, Set Trap/Conditional display, Trace options, and Zap FCB.

6502X - \$150. A 6502 cross assembler. Runs on the Z80 but assembles 6502 instructions into 6502 object code! Similar features as our Macro assemblers.

QD - \$200. A SUPER FAST Z80 assembler. Up to 10 times faster than conventional assemblers. Directly generates code into memory in one pass but also to offset for execution in its own memory space. Pascal-like structures: repeat...until...if...then...else, while...do...begin...end...case...of... Multiple statements per line, special register handling expressions, long symbol names, auto and modular assembly, and more! This one uses Zilog Mnemonics.

QED - \$150. A screen editor which is both FAST and easy to learn. Commands include block delete, copy, and move to a named file or within text, repeat previous command, change, locate, find at start of line, and numerous cursor and window movement functions. Works with any CRT having clear screen, addressable cursor, clear to end of line, clear to end of screen, and 80x24.

DISK FORMATS

When ordering software specify which disk format you would like.

CODE	DESCRIPTION
8SD	8" IBM 3740 Single Density (128 bytes/26 sectors/77 tracks)
8DD	8" Double Density (256 bytes/26 sectors/77 tracks)
8XD	8" CDL Extended Density (1024 bytes/8 sectors/77 tracks = 616K)
5SD	5.25" Single Density (TRS80 Model I, Versafloppy I, Tarbell I)
5EP	5.25" Epson Double Density
5PC	5.25" IBM PC Double Density
5XE	5.25" Xerox 820 Single Density
5OS	5.25" Osborne Single Density
5ZA	5.25" Z80 Apple (Softcard compatible)

TPM INFO When ordering TPM I or II, in addition to Disk Format, please specify one of the following codes:

CODE	DESCRIPTION
NSSD/H	North Star Single Density for Horizon I/O
NSSD/Z	North Star Single Density for Zapple I/O
NSDD/H	North Star Double Density for Horizon I/O
NSDD/Z	North Star Double Density for Zapple I/O
TRS80-I	TRS-80 Model I (4200H Offset)
TRS80II	TRS-80 Model II
V18	Versafloppy I 8"
V15	Versafloppy I 5.25"
VII8	Versafloppy II 8" (XD)
VII5	Versafloppy II 5.25"
TRS80II	TRS-80 Model II (XD)

Prices and Specifications subject to change without notice.

TPM, Z80, CP/M, TRS80 are trademarks of CDL, Zilog, DRI and Tandy respectively.

ZTEL - \$80. An extensive text editing language and editor modelled after DEC's TECO.

ZEDIT - \$50. A mini text editor. Character/line oriented. Works well with hardcopy terminals and is easy to use. Includes macro command capability.

TOP I - \$80. A Text Output Processor for formatting manuals, documents, etc. Interprets commands which are entered into the text by an editor. Commands include: justify, page number, heading, subheading, centering, and more.

TOP II - \$100. A superset of TOP I. Adds: embedded control characters in the file, page at a time printing, selected portion printing, include/merge files, form feed/CRLF option for paging, instant start up, and final page ejection.

ZDDT - \$40. This is the disk version of our famous Zapple monitor. It will also load hex and relocatable files.

ZAPPLE SOURCE - \$80. This is the source to the SMB ROM version of our famous Zapple monitor. It can be used to create your own custom version or as an example of the features of our assemblers. Must be assembled using one of our assemblers.

MODEM - A communication program for file transfer between systems or using a system as a terminal. Based on the user group version but modified to work with our SMB board or TRS-80 Models I or II. You must specify which version you want.

MODEM SOURCE - \$40. For making your own custom version. Requires one of our Macro Assemblers.

DISASSEMBLER - \$80. Does bulk disassembly of object files creating source files which can be assembled by one of our assemblers.

HARDWARE

S-100 — **SMB II Bare Board** \$50. "System Monitor Board" for S-100 systems. 2 serial ports, 2 parallel ports, cassette interface, 4K memory (ROM, 2708 EPROM, 2114 RAM), and power on jump. When used with Zapple ROM below, it makes putting a S-100 system together a snap.

Zapple ROM \$35. Properly initializes SMB I/II hardware, provides a powerful debug monitor.

IBM PC — **Big Blue Z80 board** \$595. Add Z80 capability to your IBM Personal Computer. Runs CP/M programs but does not require CP/M or TPM. Complete with Z80 CPU, 64K add on memory, serial port, parallel port, time and date clock with battery backup, hard disk interface, and software to attach to PC DOS and transfer programs. Mfr'd by QCS.

50% Discount on all CDL software ordered at the same time as a Big Blue (and for the Big Blue).

APPLE II — **Chairman Z80** \$345. Add Z80 capability to your Apple II/II Plus computer. Runs CP/M programs with our more powerful TPM. Includes 64K memory add on (unlike the competition this is also useable by the 6502/DOS as well as the Z80, TPM, QD assembler, QED Screen Editor, BUSINESS BASIC. Mfr'd by AMT Research.

Apple Special \$175. Buy the Apple Z80 Developer at the same time as the "Chairman" and pay only \$175 instead of \$325.

APPLE Z80 DEVELOPER

Includes: 6502X (\$150), MACRO II (\$100), MACRO III (\$150), QD (\$200), QED (\$150), LINKER (\$80), DEBUG I (\$80), DEBUG II (\$100), ZDDT (\$40) and BUSINESS BASIC (\$200)

VALUE: \$1250

NOW \$325

\$175 when purchased with AMT "Chairman" Board

ORDERING INFORMATION:

VISA/MasterCard/C.O.D.

Call or Write With Ordering Information....



OEMs:

Many CDL products are available for licensing to OEM's. Write to Carl Galletti with your requirements.

Dealer Inquiries Invited.



For Phone Orders ONLY Call Toll Free...

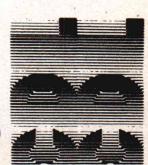
1-(800) 458-3491

(Except Pa.)

Ask For Extension #15

For information and Tech Queries call

(609) 599-2146



Circle no. 11 on reader service card.

CP/M on the Commodore 64 (Listing Continued, text begins on page 14)

Listing Two

00077	C028	AE 06 C0	LDX DISK	DISK ADDRESS
00078	C02B	A8	TAY	
00079	C02C	20 BA FF	JSR SETLFS	KERNEL
00080	C02F	A9 01	LDA #1	ONE CHARACTER NAME
00081	C031	A2 56	LDX #<FILNAM	LO ADDRESS
00082	C033	A0 C1	LDY #>FILNAM	HI ADDRESS
00083	C035	20 BD FF	JSR SETNAM	KERNEL
00084	C038	20 C0 FF	JSR OPEN	KERNEL
00085	C03B		;	
00086	C03B		;	SETUP FOR READ/WRITE LOOPS
00087	C03B		;	
00088	C03B	A9 30	LDA #ZERO	FIRST SECTOR NUMBER
00089	C03D	8D 4A C1	STA USRMSG+9	RESET CMD MSG
00090	C040	A9 05	LDA #5	FIVE SECTORS
00091	C042	8D 54 C1	STA LOOPCT	INITIALIZE LOOP CTR
00092	C045	AD 57 C1	LDA INOUT	IN OR OUT?
00093	C048	F0 06	BEQ RD	IN - GO TO READ
00094	C04A	20 B0 C0	JSR WRITIT	CALL WRITE SUBPROG
00095	C04D	4C 53 C0	JMP EXIT	
00096	C050	20 5F C0	RD	JSR READIT CALL READ SUBPROG
00097	C053		;	
00098	C053		;	WRAP UP AND RETURN
00099	C053		;	
00100	C053	A9 02	EXIT	LDA #DATCHN DATA CHANNEL
00101	C055	20 C3 FF		JSR CLOSE KERNEL
00102	C058	A9 0F		LDA #CMDCHN COMMAND CHANNEL
00103	C05A	20 C3 FF		JSR CLOSE KERNEL
00104	C05D	58		CLI INTS BACK ON
00105	C05E	60		RTS MAIN PROG EXIT
00107	C05F		;	*** READ FROM DISK ***
00108	C05F		;	
00109	C05F		;	
00110	C05F		;	READ LOOP - READ 5 SECTORS
00111	C05F		;	
00112	C05F	A9 31	READIT	LDA #ONE U1 IS AN INPUT CMD
00113	C061	8D 42 C1		STA USRMSG+1 PUT IN THE 1
00114	C064		;	
00115	C064			XAD BUF,FREKZ1 BUF ADDR FOR INLP
00121	C06C		READ1	XAD BPMMSG,FREKZ2 DISK PTR RESET CMD
00127	C074	A9 08		LDA #BFML MSG LENGTH
00128	C076	20 01 C1		JSR CMDOUT SEND IT
00129	C079			XAD USRMSG,FREKZ2 DISK READ CMD
00135	C081	A9 0B		LDA #USRML MSG LENGTH
00136	C083	20 01 C1		JSR CMDOUT SEND IT
00137	C086		;	
00138	C086	20 1B C1		JSR INLP BRING IN THE DATA
00139	C089		;	
00140	C089	EE 4A C1		INC USRMSG+9 POINT TO NEXT SECTOR
00141	C08C	E6 FC		INC FREKZ1+1 POINT TO NEXT PAGE
00142	C08E	CE 54 C1		DEC LOOPCT COUNT DOWN TO ZERO
00143	C091	DD D9		BNE READ1 NOT DONE
00144	C093		;	
00145	C093		;	MOVE "CPM" TO BASIC TEXT AREA
00146	C093		;	
00147	C093			XAD BTSTR, FREKZ1 SETUP TO MOVE BOOT65
00153	C098			XAD BOOT65, FREKZ2 TO NORMAL AREA
00159	COA3	A0 00		LDY #0 INITIALIZE LOOPCTR
00160	COA5	B1 FB	INMOV	LDA (FREKZ1),Y GET BYTE

```

00161 COA7 91 FD STA (FREKZ2),Y ;MOVE TO WORK AREA
00162 COA9 C8 INY ;LOOP CTR UP
00163 COAA 98 TYA ;INTO A FOR COMPARE
00164 COAB C9 FC CMP #BOOTLN ;MOVED ALL?
00165 COAD D0 F6 BNE INMOV ;NO-DO NEXT
00166 COAF 60 RTS ;ALL DONE - LEAVE
00168 COBO ; *** WRITE TO DISK ***
00169 COBO ;
00170 COBO ;
00171 COBO ; LOOP TO MOVE "CPM" BACK TO BUF
00172 COBO ;
00173 COBO WRITIT XAD BOOT65,FREKZ1 ;SETUP TO MOVE BOOT
00179 COB8 XAD BTSTRT,FREKZ2 ; BACK TO BUFFER
00185 COCO A0 00 LDY #0 ;INIT LOOP CTR
00186 COC2 B1 FB OUTMOV LDA (FREKZ1),Y ;GET BYTE
00187 COC4 91 FD STA (FREKZ2),Y ;PUT IN OUTBUF
00188 COC6 C8 INY ;LOOP CTR UP
00189 COC7 98 TYA ;INTO A FOR COMPARE
00190 COC8 C9 FC CMP #BOOTLN ;ALL DONE?
00191 COCA D0 F6 BNE OUTMOV ;NO-DO NEXT
00192 COCC ;
00193 COCC ; WRITE LOOP - WRITE 5 SECTORS
00194 COCC ;
00195 COCC A9 32 LDA #TWO ;U2 IS AN OUTPUT CMD
00196 COCE 8D 42 C1 STA USRMSG+1 ;PUT IN THE 2
00197 COD1 ;
00198 COD1 XAD BUF,FREKZ1 ;BUF ADDR FOR OUTLP
00204 COD9 WRIT1 XAD BFMMSG,FREKZ2 ;DISK PTR RESET CMD
00210 COE1 A9 08 LDA #BFML ;MSG LENGTH
00211 COE3 20 01 C1 JSR CMDOUT ;SEND IT
00212 COE6 ;
00213 COE6 20 2E C1 JSR OUTLP ;OUTPUT THE DATA
00214 COE9 ;
00215 COE9 XAD USRMSG,FREKZ2 ;DISK WRITE CMD
00221 COF1 A9 0B LDA #USRML ;MSG LENGTH
00222 COF3 20 01 C1 JSR CMDOUT ;SEND IT
00223 COF6 ;
00224 COF6 EE 4A C1 INC USRMSG+9 ;POINT TO NEXT SECTOR
00225 COF9 E6 FC INC FREKZ1+1 ;POINT TO NEXT PAGE
00226 COFB CE 54 C1 DEC LOOPCT ;COUNT DOWN
00227 COFE D0 D9 BNE WRIT1 ;NOT DONE YET
00228 C100 60 RTS ;ALL DONE - LEAVE
00230 C101 ;
00231 C101 ; SUBROUTINE TO OUTPUT CMD MSG
00232 C101 ;
00233 C101 8D 55 C1 CMDOUT STA MSGL ;SAVE LENGTH
00234 C104 A2 0F LDX #CMDCHN ;COMMAND CHANNEL
00235 C106 20 C9 FF JSR CHKOUT ;OPEN FOR OUTPUT
00236 C109 A0 00 LDY #0 ;POINTER TO CHAR
00237 C10B AE 55 C1 LDX MSGL ;NUM OF CHARS TO SEND
00238 C10E B1 FD CMDLP LDA (FREKZ2),Y ;GET A CHARACTER
00239 C110 20 D2 FF JSR CHRROUT ;SEND IT
00240 C113 C8 INY ;NEXT CHAR
00241 C114 CA DEX ;LOOP CTR DOWN
00242 C115 D0 F7 BNE CMDLP ;NOT DONE - NEXT
00243 C117 20 CC FF JSR CLRCHN ;KERNEL
00244 C11A 60 RTS
00245 C11B ;
00246 C11B ; SUBROUTINE TO READ 256 BYTES
00247 C11B ;
00248 C11B A2 02 INLP LDX #DATCHN ;DATA CHANNEL #
00249 C11D 20 C6 FF JSR CHKIN ;OPEN FOR INPUT
00250 C120 A0 00 LDY #0 ;BUFFER INDEX & LOOP CT
00251 C122 20 CF FF INLP1 JSR CHRIN ;GET A CHAR

```

(Continued on next page)

CP/M on the Commodore 64

(Listing Continued, text begins on page 14)

Listing Two

```
00252 C125 91 FB STA (FREKZ1),Y ;PUT IN BUFFER
00253 C127 C8 INY ;NEXT SLOT
00254 C128 D0 F8 BNE INLP1 ;256 CHARS IN A SECTOR
00255 C12A 20 CC FF JSR CLRCHN ;KERNEL CLEAR CHANNEL
00256 C12D 60 RTS
00257 C12E ;
00258 C12E ; SUBROUTINE TO WRITE 256 BYTES
00259 C12E ;
00260 C12E A2 02 OUTLP LDX #DATCHN ;DATA CHANNEL #
00261 C130 20 C9 FF JSR CHKOUT ;OPEN FOR OUTPUT
00262 C133 A0 00 LDY #0 ;BUFFER INDEX & LOOP CT
00263 C135 B1 FB OUTLP1 LDA (FREKZ1),Y ;GET A CHAR
00264 C137 20 D2 FF JSR CHRROUT ;SEND IT
00265 C13A C8 INY ;NEXT CHAR
00266 C13B D0 F8 BNE OUTLP1 ;256 CHARS IN A SECTOR
00267 C13D 20 CC FF JSR CLRCHN ;KERNEL CLEAR CHANNEL
00268 C140 60 RTS
00270 C141 ;
00271 C141 ; DATA
00272 C141 ;
00273 C141 55 31 3A USRMSG .BYTE 'U1:' ;U1:2 0 1 1
00274 C144 32 .BYTE ZERO+DATCHN
00275 C145 20 30 .BYTE ' 0 1 1'
00276 C14B 0D .BYTE CR
00277 C14C USRML =*-USRMSG
00278 C14C 42 2D BPMMSG .BYTE 'B-P:' ;B-P:2 0
00279 C150 32 .BYTE ZERO+DATCHN
00280 C151 20 30 .BYTE ' 0'
00281 C153 0D .BYTE CR
00282 C154 BPMI =*-BPMMSG
00283 C154 00 LOOPCT .BYTE 0
00284 C155 00 MSGL .BYTE 0
00285 C156 23 FILNAM .BYTE '*'
00286 C157 00 INOUT .BYTE 0 ;IN=0, OUT=1
00287 C158 ;
00288 C158 .END
```

ERRORS = 00000

SYMBOL TABLE

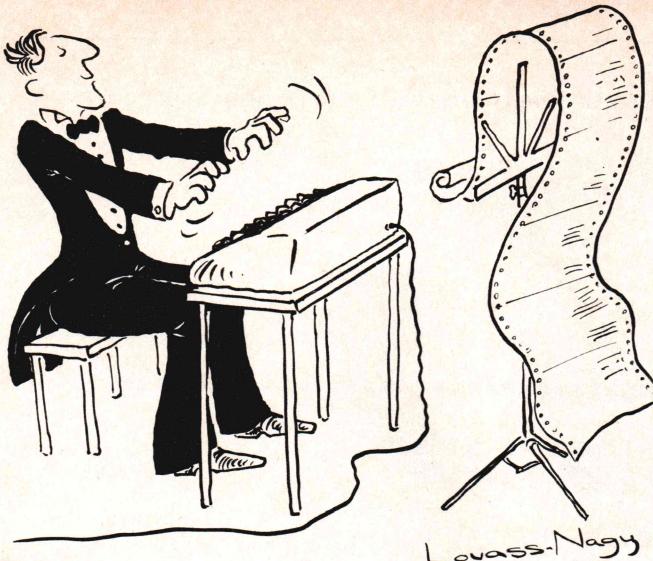
SYMBOL VALUE

BIOS65	0A00	BOOT65	0801	BOOTLN	00FC	BPMI	0008
BPMMSG	C14C	BTSTRT	0904	BUF	0900	CHKIN	FFC6
CHKOUT	FFC9	CHRIN	FFCF	CHRROUT	FFD2	CLOSE	FFC3
CLRCHN	FFCC	CMDCHN	000F	CMDLP	C10E	CMDOUT	C101
CR	000D	DATCHN	0002	DISK	C006	EXIT	C053
FILNAM	C156	FREKZ1	00FB	FREKZ2	00FD	INLP	C11B
INLP1	C122	INMOV	C0A5	INOUT	C157	LOOPCT	C154
MSGL	C155	ONE	0031	OPEN	FFC0	OUTLP	C12E
OUTLP1	C135	OUTMOV	C0C2	RD	C050	READ	C007
READ1	C06C	READIT	C05F	SETLFS	FFBA	SETNAM	FFBD
START	C014	TWO	0032	USRML	000B	USRMSG	C141
WRIT1	C0D9	WRITE	C00F	WRITIT	C0B0	XAD	FFFF
ZERO	0030						

END OF ASSEMBLY

End Listing Two

(Listing Three begins on page 26)



Lou Ann Nagy

Before Johann Sebastian Bach developed a new method of tuning, you had to change instruments practically every time you wanted to change keys. Very difficult.

Before Avocet introduced its family of cross-assemblers, developing micro-processor software was much the same. You needed a separate development system for practically every type of processor. Very difficult and very expensive.

But with Avocet's cross-assemblers, a single computer can develop software for virtually any microprocessor! Does that put us in a league with Bach? You decide.

The Well-Tempered Cross-Assembler

Development Tools That Work

Avocet cross-assemblers are fast, reliable and user-proven in over 3 years of actual use. Ask NASA, IBM, XEROX or the hundreds of other organizations that use them. Every time you see a new microprocessor-based product, there's a good chance it was developed with Avocet cross-assemblers.

Avocet cross-assemblers are easy to use. They run on any computer with CP/M® and process assembly language for the most popular microprocessor families.

5 1/4" disk formats available at no extra cost include Osborne, Xerox, H-P, IBM PC, Kaypro, North Star, Zenith, Televideo, Otron, DEC.

Turn Your Computer Into A Complete Development System

Of course, there's more. Avocet has the tools you need from start to finish to enter, assemble and test your software and finally cast it in EPROM:

Text Editor VEDIT -- full-screen text editor by CompuView. Makes source code entry a snap. Full-screen text editing, plus TECO-like macro facility for repetitive tasks. Pre-configured for over 40 terminals and personal computers as well as in user-configurable form.

CP/M-80 version \$150
CP/M-86 or MDOS version \$195
(when ordered with any Avocet product)

EPROM Programmer -- Model 7128 EPROM Programmer by GTek programs most EPROMS without the need for personality modules. Self-contained power supply ... accepts ASCII commands and data from any computer through RS 232 serial interface. Cross-assembler hex object files can be down-loaded directly. Commands include verify and read, as well as partial programming.

PROM types supported: 2508, 2758, 2516, 2716, 2532, 2732, 2732A, 27C32, MCM8766, 2564, 2764, 27C64, 27128, 8748, 8741, 8749, 8742, 8751, 8755, plus Seeq and Xicor EEPROMS.

Avocet Cross-assembler	Target Microprocessor	CP/M-80 Version	CP/M-86 IBM PC, MSDOS** Versions
XASMZ80	Z-80		
XASM85	8085		
XASM05	6805		
XASM09	6809		
XASM18	1802		
XASM48	8048/8041		
XASM51	8051		
XASM65	6502		
XASM68	6800/01		
XASMZ8	Z8		
XASMF8	F8/3870		
XASM400	COP400		
XASM75	NEC 7500		
Coming soon: XASM68K...68000			\$500.00
		\$200.00 each	
			\$250.00 each
			\$300.00 each

(Upgrade kits will be available for new PROM types as they are introduced.)

Programmer \$429
Options include:

Software Driver Package -- enhanced features, no installation required.

CP/M-80 Version \$ 75
IBM PC Version \$ 95
RS 232 Cable \$ 30
8748 family socket adaptor ... \$ 98
8751 family socket adaptor ... \$174
8755 family socket adaptor ... \$135

G7228 Programmer by GTek -- baud to 2400 ... superfast, adaptive programming algorithms ... programs 2764 in one minute.

Programmer \$549

Ask us about Gang and PAL programmers.

HEXTRAN Universal HEX File Converter -- Converts to and from Intel, Motorola, MOS Technology, Mostek, RCA, Fairchild, Tektronix, Texas Instruments and Binary formats.

Converter, each version \$250

Call Us

If you're thinking about development systems, call us for some straight talk. If we don't have what you need, we'll help you find out who does. If you like, we'll even talk about Bach.

CALL TOLL FREE 1-800-448-8500
(In the U.S. except Alaska and Hawaii)

VISA and Mastercard accepted. All popular disc formats now available -- please specify. Prices do not include shipping and handling -- call for exact quotes. OEM INQUIRIES INVITED.

*Trademark of Digital Research **Trademark of Microsoft

AVOCET SYSTEMS INC.™
DEPT. 684-DDJ
804 SOUTH STATE STREET
DOVER, DELAWARE 19901
302-734-0151 TELEX 467210



Listing Three

CPM232.TX.

LINE#	LOC	CODE	LINE	
00001	0000		*****	
00002	0000		;	
00003	0000		; MODIFICATIONS TO BIOS65	
00004	0000		FOR AN RS-232 PRINTER	
00005	0000		;	
00006	0000		; W.G. PIOTROWSKI	
00007	0000		;	
00008	0000		*****	
00009	0000		;	
00010	0000		; EQUATES	
00011	0000		;	
00012	0000	FILE =128	;FILE NUMBER	
00013	0000	BUF232 =\$F00	;NEW RS232 BUF LOC	
00014	0000	CHAR =\$901	;CHAR BEING PRINTED	
00015	0000	;		
00016	0000	SETLFS =\$FFBA	;KERNEL ROUTINES	
00017	0000	SETNAM =\$FFBD	;	
00018	0000	OPEN =\$FFC0	;	
00019	0000	CLOSE =\$FFC3	;	
00020	0000	CHKOUT =\$FFC9	;	
00021	0000	CLRCHN =\$FFCC	;	
00022	0000	CHROUT =\$FFD2	;	
00023	0000	;		
00024	0000	ENABL =\$2A1	;RS232 STILL ACTIVE	
00025	0000	ROBUF =\$F9	;RS232 OUTBUF PTR	
00026	0000	RIBUF =\$F7	;RS232 INBUF PTR	
00027	0000	;		
00028	0000	*=\$A9F		
00029	0A9F	;		
00030	0A9F	AE E2 0A	LDX ENTFLG	;SEE IF FIRST ENTRY
00031	0AA2	DO 26	BNE OUTCHR	;ZERO - OPEN FILE
00032	0AA4	;		
00033	0AA4	;	INITIAL ENTRY	
00034	0AA4	;	OPEN FILE AND MOVE BUFFER	
00035	0AA4	;		
00036	0AA4	A2 01	OPENIT LDX #1	;NOT ZERO MEANS ENTERED
00037	0AA6	8E E2 0A	STX ENTFLG	;PUT IN FLAG
00038	0AA9	A9 80	LDA #FILE	;FILE NUMBER
00039	0AAB	A2 02	LDX #2	;RS-232 DEVICE
00040	0AAD	A0 FF	LDY #\$FF	;NO COMMAND
00041	0AAF	20 BA FF	JSR SETLFS	;CALL KERNEL
00042	0AB2	A9 02	LDA #2	;TWO CHAR NAME
00043	0AB4	A2 E3	LDX #<SET232	;LO ADDRESS
00044	0AB6	A0 0A	LDY #>SET232	;HI ADDRESS
00045	0AB8	20 BD FF	JSR SETNAM	;KERNEL
00046	0ABB	20 C0 FF	JSR OPEN	;KERNEL
00047	0ABE	A2 00	LDX #<BUF232	;BUFFER LO ADDR
00048	0AC0	A0 0F	LDY #>BUF232	;BUFFER HI ADDR
00049	0AC2	86 F9	STX ROBUF	;KERNEL OUT BUF ADDR
00050	0AC4	84 FA	STY ROBUF+1	;HI ORDER PART
00051	0AC6	86 F7	STX RIBUF	;MOVE INBUF TOO
00052	0AC8	84 F8	STY RIBUF+1	;JUST IN CASE
00054	0ACA	;		
00055	0ACA	;	CHARACTER OUTPUT	

00056	OACA		;		
00057	OACA	A2 80	OUTCHR	LDX #FILE	;GET FILE NUM
00058	OACC	20 C9 FF		JSR CHKOUT	;OPEN FOR OUTPUT
00059	OACF	B0 D3		BCS OPENIT	;ERROR MEANS CLOSED
00060	OAD1	AD 01 09		LDA CHAR	;GET CHARACTER AGAIN
00061	OAD4	20 D2 FF		JSR CHROUT	;KERNEL
00062	OAD7	AD A1 02	WAIT	LDA ENABL	;GET STATUS
00063	OADA	29 01		AND #1	;STILL RUNNING BIT
00064	OADC	D0 F9		BNE WAIT	;HANG UNTIL DONE
00065	OADE	20 CC FF		JSR CLRCHN	;CLEAR CHANNEL
00066	OAЕ1	60		RTS	
00067	OAЕ2		;		
00068	OAЕ2	00	ENTFLG	.BYTE 0	;FIRST ENTRY FLAG
00069	OAЕ3	86	SET232	.BYTE \$86,0	;RS232 PARAMS
00069	OAЕ4	00			
00070	OAЕ5		;		
00071	OAЕ5			END	

ERRORS = 00000

SYMBOL TABLE

SYMBOL	VALUE							
BUF232	0F00	CHAR	0901	CHKOUT	FFC9	CHRROUT	FFD2	
CLOSE	FFC3	CLRCHN	FFCC	ENABL	02A1	ENTFLG	0AE2	
FILE	0080	OPEN	FFC0	OPENIT	0AA4	OUTCHR	0ACA	
RIBUF	00F7	ROBUF	00F9	SET232	0AE3	SETLFS	FFBA	
SETNAM	FFBD	WAIT	0AD7					

END OF ASSEMBLY

End Listing Three

MOPI™

The Define Your Own Instructions Assembler/Compiler (Universal Cross Assembler)

MOPI is a unique software development system which combines familiar and time-proven processes to create a new and truly effective approach to the challenge of software development. The functions of an assembler and compiler are combined into one efficient, streamlined process with many added special features and an extensive set of utility programs.

MOPI enables its users to develop their own customized languages, to meet the needs of any application, and to develop programs for any 8-bit microprocessor.

It is simple enough for a beginning programmer, yet sophisticated enough to meet the needs of the most highly trained and experienced computer professional. MOPI is flexible enough to be adapted and enhanced as the user's needs change or as Technology advances.

When standard assembly language is too difficult, time-consuming, or just too old fashioned, MOPI's user-defined instruction capabilities add flexibility and efficiency.

MOPI software: \$175 Manual: \$30

Currently available for CP/M with 64K memory.

Inquire concerning availability on other systems.

For more information and ordering instructions, write

VOCS

P.O. Box 3705, Minneapolis, MN 55403

MOPI is a trademark of Voice Operated Computer Systems

CP/M is a trademark of Digital Research Inc.

RP/M

T.M.

dBASE II Programming Techniques

It seems that everyone nowadays is using a mailing list! Lawyers track clients, clubs chart members, and businesses of all types garner lists of names for mass mailings. I handle a small 1200-name mailing list for a missionary friend who sends field reports to his supporters every month.

Usually, a mailing list will eventually involve the Post Office, and if the list is long enough (200 pieces or more) bulk mailing can save money over regular first class mail. Ah, but there is a catch. The Postmaster expects you to pre-sort your mailing, count it, and have accurate zip

code information. Bulk mail is not generally forwarded when an incorrect zip code sends your letter to the wrong city even if the proper city and state are part of the address.

Exact zip code validation is impractical. However, there is a way to partially verify zip codes and validate the state. This technique should decrease errors and increase mass mailing efficiency and accuracy. This procedure could be written in high-level language, but for speed and compactness of code I chose to write it as a machine language subroutine.

ZIP-CHK.ASM assembles into a machine code subroutine that uses a table published by the Post Office to first verify that the two-character state abbreviation is correct then check that the zip code is within the range for that state as set by the table. The source listing is well commented and documented. It was written as a dBASE II subroutine but can easily

be adapted to any high-level language that can pass a string type memory variable to the called subroutine and modify that variable if required and pass it back to the calling program.

I got the idea for this program from Bertel Schmitt who wrote an earlier version as part of a file called DBASM.ASM, found on many RCP/M systems around the country. I modified and streamlined the routine, correcting a few errors in the original code. I added a symmetrical look-up table that increased accuracy and speeded the table search.

If you have a modem, you can find ZIP-CHK.ASM on several RCP/M systems, including the dBASE II system at (408) 378-8733. **DBJ**

by Gene Head

Gene Head, 2860 N.W. Skyline Dr., Corvallis, OR 97330.

dBASE II Techniques Listing

```
*****
***** ZIP-CHK.ASM *****
*****
```

This machine-language subroutine is designed to be called by a high-level language such as dBASE II or BASIC with a single passed variable of exactly five characters like 'ST123', where ST is the Post Office designated two letter abbreviation for the state and 123 are the first three ZIP code digits of that state.

On return from this sub-routine, the passed variable is un-changed if the ZIP code is valid for the state. If the two letter abbreviation is invalid then the returned variable becomes 'ERR 1' or if the state is valid but the ZIP code is out of range then the variable becomes 'ERR 2'.

The following command program could be used to verify a state and ZIP code in a dBASE II data file:

```
*      ZIP-CHK.CMD
*
* Note:  The following two lines of commands should be
*        a part of the main program initialization
*
*      LOAD ZIP-CHK
*      SET CALL TO 42096
*
STORE $(STATE,1,2)+$(ZIP,1,3) TO ZIP:CHK
CALL ZIP:CHK
```

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 193.

```
IF 'ERR'$ZIP:CHK  
DO ERROR:HANDLING  
ENDIF  
  
RETURN
```

Adapted from DBASM.ASM file by Bertel Schmitt (c) 1983

Modified and expanded and accuracy enhanced by:

Gene Head
Head Quarters
2860 NW Skyline Drive
Corvallis, OR 97330
(503) 758-0279

Last up-date 02-02-84
02-26-84

START EQU 42096 ; DECIMAL LOCATION OF SUB-ROUTINE
ENDTAB EQU OFFH ; END OF TABLE MARKER

```
; ERROR CODES
NOSTA EQU      '1'      ;NO SUCH STATE FOUND
WRZIP  EQU      '2'      ;BAD ZIP - DOES NOT MATCH STATE
```

ORG START

```
BEGIN:  INX      H      ; skip length byte of passed variable
        MOV      D,M    ; put state in DE
        INX      H
        MOV      E,M    ; and save a pointer
        SHLD    ZIPPNT  ; ...to the ZIP bytes
```

; First try to locate the state in the table

LXI B,B : bumper (each table entry is 8 bytes)
LXI H, TABLE : point to first table entry

```

LOOP1: MOV    A,M    ; get table character
        CPI    ENDTAB ; see if the end of the table has
        JZ     ERROR1 ; ... been found and error out if true
        CMP    D       ; Otherwise try to match first state character
        JZ     MAYBY  ; If first character matches we MAY have out state
        DAD    B       ; Otherwise BUMP the table pointer to the next
        JMP    LOOP1  ; entry and try again, and again, and again.....

```

Here if first state character matches so try for second match

```

MAYBY INX H      ; Fetch second state character
        MOV A,M
        CMP E      ; Check it against passed variable
        JZ MATCH  ; ZERO will be set if valid state was found
        DCX H      ; Otherwise back up one to keep BUMPER
        DAD B      ; ...in SYNC and BUMP to next table entry.
        JMP LOOP1  ; Try again, and again, and again...

```

(Continued on next page)

dBASE II Techniques Listing (Listing Continued, text begins on page 28)

; Here when valid state was found so check if ZIP is within range

```
MATCH: PUSH H ; Save pointer to TABLE ZIPS
       LHLD ZIPPNT ; HL points to passed ZIP
       MOV B,H ; BC now points to passed ZIP
       MOV C,L ;....and
       POP H ;.....HL points to TABLE ZIPS
```

; First check if passed ZIP is smaller than first TABLE ZIP

```
       CALL RANGE ; Check first digit
       JC ERROR2 ; Carry set if too small
       CALL RANGE ; Check second digit
       JC ERROR2 ; Carry set if too small
       CALL RANGE ; Check last digit ONLY if. . .
       JC ERROR2 ; . . . first two were exact matches
```

DCX B ; Keep bumper is sync
MATCH1 DCX B
 DCX B

```
       PUSH B ; Swap compared elements
       PUSH H ; so now we check for upper range
       POP B
       POP H
```

* UPPER LIMIT CHECK

```
       CALL RANGE ; Check first digit
       JC ERROR2 ; Error out if too large
       CALL RANGE ; Check second digit and error
       JC ERROR2 ; ...out if it is too large
       RNZ ; If zero then test is passed so return
       ; ...to calling program
       CALL RANGE ; Check last digit ONLY if . .
       JC ERROR2 ; . . .first two were exact matches
```

```
       RET ; RETURN to calling program all OK
```

* RANGE CHECKER

```
RANGE: INX B ; Compare NEXT byte
       INX H
       LDAX B ; Set CARRY if out of range
       CMP M
       RET
```

; ERRORHANDLERS

```
ERROR1: LHLD ZIPPNT
       MVI A,'1'
       JMP ERROR
```

```
ERROR2: LHLD ZIPPNT
       MVI A,'2'
```

* REPLACE VARIABLE WITH ERROR MESSAGE

```
ERROR: LHLD ZIPPNT
       DCX H
       MVI M,'E'
```

```

INX      H
MVI      M, "R"    ; AND NOW ERR
INX      H
MVI      M, "R"
INX      H
MVI      M, " "
INX      H
MOV      M,A
RET      ; Return to calling program
          ; ...with passed variable set to ERROR
ZIPPNT: DS      2

```

• The following info is copied from the US Zipcode book.
 • Small possessions and islands are not included but may be added.

TABLE:

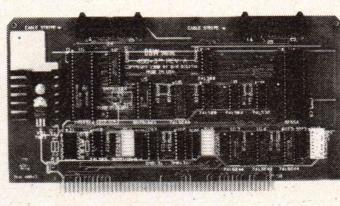
DB	"AL350369"	DB	"NV890898"
DB	"AK995999"	DB	"NH030038"
DB	"AZ850865"	DB	"NJ070089"
DB	"AR716729"	DB	"NM870884"
DB	"CA900961"	DB	"NY100149"
DB	"CO800816"	DB	"NC270289"
DB	"CT060069"	DB	"ND580588"
DB	"DE197199"	DB	"OH430458"
DB	"DC200205"	DB	"OK730749"
DB	"FL320339"	DB	"OR970979"
DB	"GA300319"	DB	"PA150196"
DB	"HI967968"	DB	"RI028029"
DB	"ID932938"	DB	"SC290299"
DB	"IL600629"	DB	"SD570577"
DB	"IN460479"	DB	"TN370385"
DB	"IA500528"	DB	"TX750799"
DB	"KS660679"	DB	"UT840847"
DB	"KY400427"	DB	"VT050059"
DB	"LA700714"	DB	"VA220246"
DB	"ME039049"	DB	"WA980994"
DB	"MD206219"	DB	"WV247268"
DB	"MA010027"	DB	"WI530549"
DB	"MI480499"	DB	"WY820831"
DB	"MN550567"	DB	ENDTAB
DB	"MS386397"	DB	END
DB	"MO630658"		
DB	"MT590599"		
DB	"NE680693"		

End Listings



**THE
488+3 IEEE 488 TO S-100 INTERFACE**

IEEE-488



S-100

- Handles all IEEE-488 1975/78 functions
- IEEE 696 (S-100) compatible
- MBASIC subroutines supplied; no BIOS mods required
- 3 parallel ports (8255A-5)
- Industrial quality; burned in and tested
- \$375

[Dealer inquiries invited]

D&W DIGITAL
20655 Hathaway Ave.
Hayward, CA 94541 415/ 887-5711

Circle no. 21 on reader service card.

First Chinese Forth A Double-Headed Approach

If the computer had been invented in China, what problems would English-speaking people have to surmount in order to use it? To learn computers, a person who comes from an Oriental culture must overcome at least two major obstacles. The first is the natural language barrier and the second is the computer language barrier. Whenever I read an article on computers that is written in English, I first have to understand the English literal meaning, then the technical aspects of the article. Even though I have had many years of English education, I still have trouble now and then. This difficulty will diminish with time, but it will never disappear totally.

With this kind of painful experience in mind (would you guys give me a break and publish articles in plain and simple English?), I started a Chinese Forth computer project, which should be completed by the time this article is published. I primarily wanted to provide a computer system that is more easily accessible to my countrymen, the Chinese. After all, we make up more than a quarter of the world's population: over 1,000,000,000 people. Since the beauty of Forth has been established time and time again, it was, of course, my first choice without reservation. The whole project was written in Dai-E Systems' Forth Level II in both Chinese and English.

The significance of Chinese Forth (or any Chinese language computer system) is obvious. In Taiwan, as well as other Chinese communities, English is a mandatory second language in all schools. But no matter how well a Chinese person masters the English language, Chinese remains the first choice. Computer languages are only software tools invented to enable us to communicate with hardware. The closer this form of expression is to one's native language, the more comfortably the user can proceed, and with much better results.

Many concepts expressed in Chinese have no English equivalent, and vice versa. This is one reason why translation is a difficult problem. Since the computer was

"My main objective is to provide the Chinese people with a simple but powerful computer language."

		Dai-E Systems, Inc.										
		HEX	8	9	A	B	C	D	E	F		
HEX	B	1000	1001	1010	1011	1100	1101	1110	1111			
	8765	B4321										
0	0000		128	144	SP	160	0	176	0	192	1	208
1	0001		129	145	!	161	1	177	2	193	2	209
2	0010		130	146	"	162	2	178	3	194	3	210
3	0011		131	147	#	163	3	179	4	195	4	211
4	0100		132	148	\$	164	4	180	5	196	5	212
5	0101		133	149	%	165	5	181	6	197	6	213
6	0110		134	150	&	166	6	182	7	198	7	214
7	0111		135	151	'	167	7	183	8	199	8	215
8	1000		136	152	(168	8	184	9	200	9	216
9	1001		137	153)	169	9	185	=	201	10	217
A	1010		138	154	*	170	:	186	<	202	11	218
B	1011		139	155	+	171	;	187	4	203	12	219
C	1100		140	156	,	172	<	188	204	13	220	235
D	1101		141	157	-	173	=	189	205	14	221	236
E	1110		142	158	.	174	>	190	206	15	222	237
F	1111		143	159	/	175	?	191	207	16	223	238
												255

Phonetic Characters

Tone Characters

Table I.

Chinese Phonetic Character Chart

by Timothy Huang

Timothy Huang, Dai-E Systems Inc.,
29783 Town Center Loop West, P. O.
Box 790, Wilsonville, OR 97070.

born in the United States, most of the current Forth-related subjects are expressed in English: SPACE, SPACES, BLANK, BLANKS, etc. For the average American, these constructions are natural, but the Chinese do not express plurals in this manner; they have a different philosophy, a different way of thinking, and different forms of expression.

Forth is expressed nicely in English, so why not in the oldest language — Chinese? For the Chinese to realize real computer power (e.g., Forth power), it should be available in their own language.

My main objective is to provide the Chinese people with a simple but powerful computer language. I want to remove the English language barrier so that even Chinese children can use Forth. Furthermore, I would like to see the impact on Forth of exposure to the Chinese culture and language. I hope to see great and positive contributions and believe that only Forth can absorb this challenge.

Other less conceptual languages will have a difficult time, because the Chinese philosophy is very abstract.

Double (Multiple) Names

A Forth word with two different names can be made easily by:

: PUD DUP;

or

:FI [COMPILE] IF ;IMMEDIATE

but there are many deficiencies in this method. For example, the execution speed will be hampered because the new word(s) must execute at least one extra cycle of the inner interpreter.

In his article entitled "Turtle Talk," Glenn Tenney described a defining word, ALIAS, that stores the old word's cfa into the new word's pfa.¹ The new word (alias) is created as an immediate word: when alias is executed, the old word (parent) must be executed. This approach solved

some problems but also created its own. In our case, not all Chinese words are immediate.

Since the implementation of Dai-E Systems' Forth Level II (83 Standard) uses a separate header and body approach, we can achieve the double (multiple) name easily by simply creating a new header with a pointer pointing to the same body of the old word. The separated heads concept was first proposed and implemented by Klans Schleisiek.² I'm not arguing the respective merits of continuous or separated heads/bodies, but only pointing out how I did it. Using the separated heads scheme, the memory map of a word is shown in Figure 1 (below).

The pointer in the above structure is the key, since nothing forbids more than one head pointing to the same body. The only no-no is to have one head pointing to many bodies, which would result in a very interesting Forth system. Equipped with this understanding, we can make

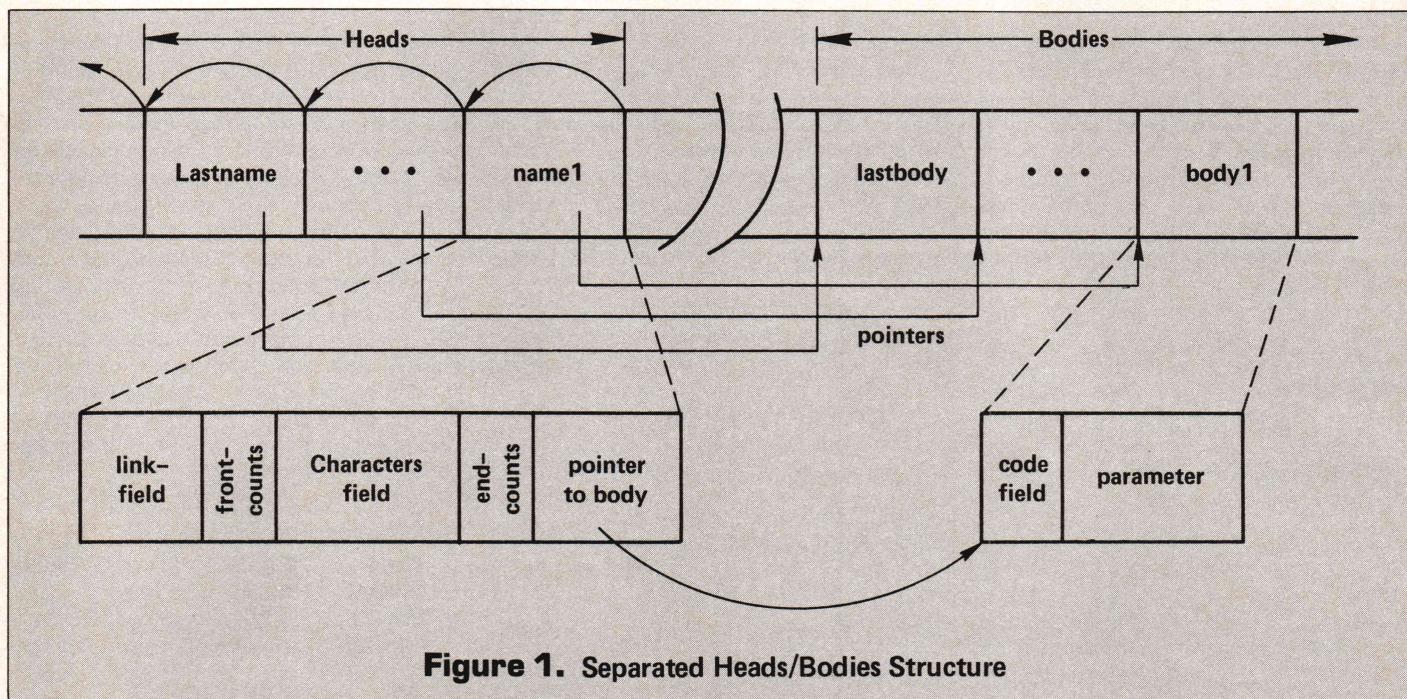


Figure 1. Separated Heads/Bodies Structure

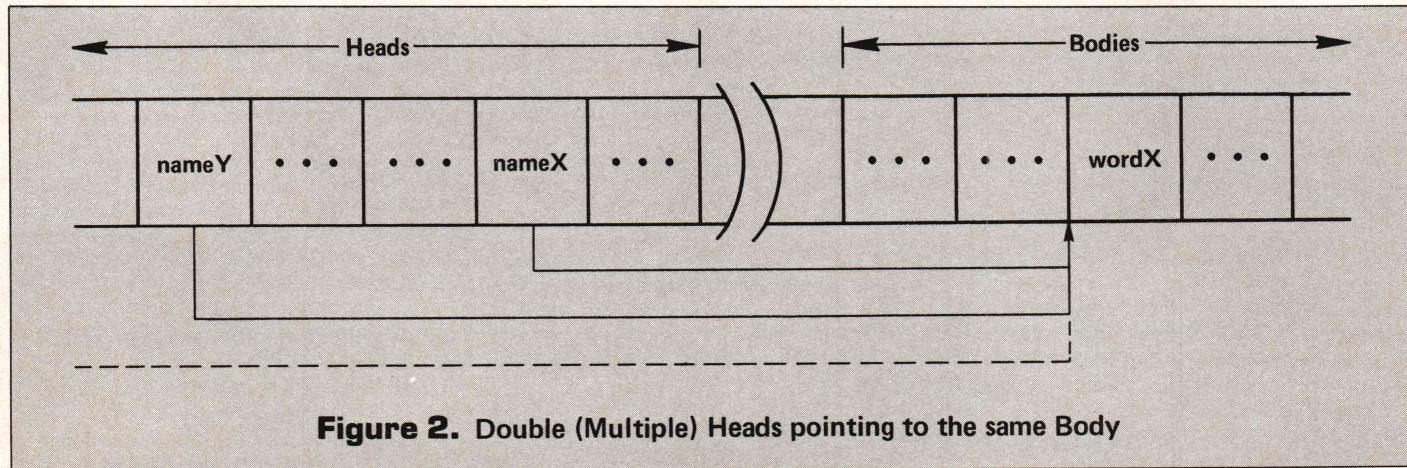


Figure 2. Double (Multiple) Heads pointing to the same Body

double (multiple) headers as shown in Figure 2 (page 33).

To accomplish this, we have to store the old pointer from the old word into the pointer location of the new name, as described in Figure 3 (below). The word C: should be used to generate new names. For example:

```
C: PUD DUP ;  
C: FI IF ; IMMEDIATE
```

The new words (actually new names) could be either immediate or non-immediate words depending on whether we issue IMMEDIATE after the definition.

This simple approach not only eliminates one extra nesting level but also gives us the choice of indicating the immediate or non-immediate nature of a word. With this trick we can have one word behaving differently depending on which header we use; i.e., one name is immediate and the other is non-immediate.

Chinese Forth

Chinese characters are ideographic symbols. Each one is an individual entity, and there are more than 50,000 of them. Because it is impossible to make an ASCII table to include all of them, we took a simpler approach and used the phonetic characters rather than the actual ideograms. There are 37 phonetic symbols and five tone symbols, as shown in Table I

(page 32). With a proper combination of at least one, but no more than three, phonetic symbols and one tone symbol, all the ideographic characters can be expressed.

These phonetic characters can be placed into the higher portion of the ASCII table with the most significant bit on. This phonetic character set is called the "BER PER MER FER" (ㄅ ㄉ ㄕ ㄕ ㄔ). This is the first thing learned when a person enters a formal education institution. The phonetic characters are learned first, then the pronunciation, then the ideographic characters. Eventually, phonetic characters (pronunciation) are no longer needed alongside each ideographic character.

As mentioned earlier, more than 50,000 ideographic characters exist now; however, there are only about 2,000 legal pronunciations (combinations of sounds and tones) for them. Thus, the homonym (different character with identical pronunciation) is a very annoying problem. Theoretically, each pronunciation can apply to about 25 different ideographic characters. However, the Chinese created a clever method for alleviating this confusion: they used multiple characters to form a phrase as often as possible rather than single characters. With phrases of at least two characters, the chance of a mix-up was greatly reduced. The telephone directory service of the Taiwan Telephone

and Telegraph Company is using this method in conjunction with their computer systems with great success.

For this reason, I decided to avoid single-character phonetic names for the Chinese Forth words. Each English Forth word was translated to a Chinese phrase of more than two characters. However, the math operators remained the same. Certain English characters used in the name fields were also preserved, such as ";" " " " " (" and so on.

Table II (page 36) lists all the required words in the 8.3 Standard. The first column is the original English, the second the equivalent Chinese ideographic characters, the third the phonetic names used in our system. Within the phonetic names, the first tone character, which is usually the blank character, was changed to a high raised "-" character to avoid the space character limitation of Forth names.

By using the phonetic characters, even though I cannot see the ideographic characters, I can pronounce the program and thus have no problem whatsoever in understanding what I have written. This computerized solution solves the 50,000-character ASCII table problem. Besides, this reflects the hope that Forth, a written language, might be a spoken language as well. Otherwise, why is there a pronunciation for words such as @ (fetch) or ! (store)? Chinese Forth not only can be written but also can be verbalized (spoken).

The other interesting thing regarding our Chinese Forth is, while decompiling the English-origin word, one may see half English and half Chinese, depending on how many English words were translated into their Chinese equivalents. This is because we utilized the double-header approach mentioned earlier.

References

1. Glenn S. Tenney. "Turtle Talk." *FORML*, 1981, pp. 521-542.
2. Klans Schleisiek. "Separated Heads." *FORTH DIMENSIONS*, Vol. II/5, p. 147.

The Dai-E Chinese Computer System includes a Victor 9000 microcomputer with 512K of RAM, Axiom IMP4 printer, and 5,000 of the most commonly used Chinese characters encoded to conform to the communication protocol selected by the U. S. Library of Congress, CCCII - Chinese Character Code for Information Interchange. In addition to their Forth system, they have a Chinese word processor. They should be able to use their system shortly on the British-made Apricot computer. They are also in the process of bringing the system up on other computers. - Ed. ■■■

```
: C: (S ----- )      \ create double header  
create  
'  
heads  
here 2-  
'  
bodies  
'  
;  
      \ make new name field  
      \ find the cfa of the old word  
      \ set dictionary pointer to header  
      \ into the new pointer address  
      \ store old cfa into new pointer address  
      \ reset dictionary pointer to body portion
```

Figure 3.
Double (Multiple) Header Code

一九八三年電腦大展
達意資訊公司
正式展出
中文電腦系統
敬請多多指教

Figure 4.
Characters printed on Axiom IMP4

(Table II begins on page 36)

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service No. 194.

Introducing

WALTZ LISP^{T.M.}

The one and only adult Lisp system for CP/M users.

Waltz Lisp is a very powerful and complete implementation of the Lisp programming language. It includes features previously available only in large Lisp systems. In fact, Waltz is substantially compatible with Franz (the Lisp running under Unix), and is similar to MacLisp. Do not be deceived by the low introductory price.

Waltz Lisp is a perfect language for Artificial Intelligence programming. It is also suitable for general applications. In fact, due to the ease of handling of textual data and random file access functions, it is often easier to write a utility program in Waltz Lisp than in any other programming language. Several general purpose utilities (including **grep** and **diff**) written entirely in Waltz Lisp are included with the interpreter.

Much faster than other microcomputer Lisps. • Long integers (up to 611 digits). Selectable radix. • True dynamic character strings. Full string operations including fast matching/extraction. • Random file access. • Binary files. • Standard CP/M devices. • Access to disk directories. • Functions of type lambda (expr), nlambda (expr), lexpr, macro. • Splicing and non-splicing character macros. • User control over all aspects of the interpreter. • Built-in prettyprinting and formatting facilities. • Complete set of error handling and debugging functions including user programmable processing of undefined function references. • Optional automatic loading of initialization file. • Powerful CP/M command line parsing. • Fast sorting/merging using user defined comparison predicates. • Full suite of mapping functions, iterators, etc. • Over 250 functions in total. • Extensive manual with hundreds of illustrative examples.

Waltz Lisp requires C/PM 2.0, Z80 and 48K RAM (more recommended). SS/SD 8" and most common 5" disk formats.

Introductory Price....\$94.50

Manual only	\$20.00
(refundable with order)	
additional charges	
\$10.00 conversion fee for 5" Diskettes	
\$3.00 C.O.D. charge	

Call toll free 1-800-LIP-4000 Ask for Dept. #3
In Oregon and outside U.S.A. call 1-503-684-3000
Unix® Bell Laboratories. CP/M® Digital Research Corp.

Circle no. 48 on reader service card.

PC
CODE^{T.M.}
INTERNATIONAL

P. O. Box 7301
Charlottesville, VA 22906

Introducing the Creative Genius...

YOU.

Discover how easy programming can be with DataBurst™.

A unique runtime screen processor and source program generator, DataBurst™ will decrease your program development time and increase the value of your application programs. The unique DataBurst™ screen editor provides fast, easy screen design. Program independent screen formats reduce both development and maintenance time.

During execution of your program, DataBurst™ controls all user interaction through one assembly language interrupt service routine, requiring as little as 14K of memory. A true full-screen processor, DataBurst™ allows unlimited design complexity, and brings a mainframe advantage to your IBM® PC.

DataBurst™ is available through your local computer retailer or directly from Key Solutions, Inc. To order directly, please send check or money order for \$225* to Key Solutions, Inc., P.O. Box 2297, Santa Clara, CA 95055. Additional language support (BASIC Compiler and C Compiler) is available for \$40*. (Please inquire about release dates for other language interfaces).

IBM® is a registered trademark of IBM Corporation.
DataBurst™ is a trademark of Key Solutions, Inc.

©Copyright 1984 Key Solutions, Inc.

The Design Tool for the Creative Programmer

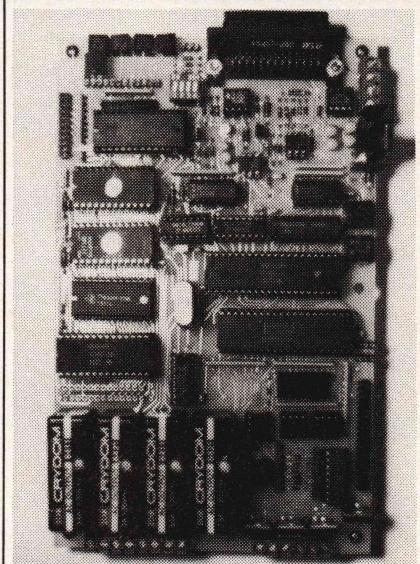


*In California add applicable sales tax.

Circle no. 31 on reader service card.

Introducing PROATROL™

The PROATROL™ single board computer is both a low cost development system for programming in **FORTH**, and a powerful target system for a wide range of real time control applications.



The 5" x 8" stand alone board is based on the Rockwell single chip FORTH Microcomputer which contains a FORTH operating system in its internal ROM.

- RS 232 and 20 ma serial interface
- 4 sockets for plug in I/O modules (Opto 22, Crydom, etc.)
- 4 half bridge power drivers
- 8 optically isolated inputs
- 8 channel 8 bit ADC, ratiometric or referenced
- 4 potentiometers for level/speed settings
- 4 position dip switch
- 4 Schmitt trigger inputs
- 4 open collector outputs
- 8 expansion I/O lines
- Power fail interrupt/reset
- In circuit EPROM and EEPROM programming
- +5v regulator, DC/DC convertor
- Buss expansion connector
- Sockets for up to 14kb of memory

PROATROL™ has it all together!

Develop your program, configure your I/O options, debug your system and GO FORTH!

Only \$649.00

To order send remittance to:

PROA Corporation
4019 EDITH BLVD. NE BLDG. 2B
ALBUQUERQUE, NM 87107
For further information call
(505) 344-2106

Circle no. 47 on reader service card.

(()) Indicates Chinese Pronunciation only

<u>English</u>	<u>Chinese Character Name</u>	<u>Chinese Forth Phonetic Name</u>
<u>Nucleus Layer</u>		
!	(儲存)	!
★	(相乘)	★
★/	(相除)	★/
+	(相加)	+
+!	(加存)	+!
-	(相減)	-
/	(相除)	/
/MOD	(餘除)	/MOD
0<	(小於零)	0<
0=	(等於零)	0=
1+	(壹加)	1+
1-	(壹減)	1-
2+	(贳加)	2+
2-	(贳減)	2-
2/	(贳除)	2/
<	(小於)	<
=	(等於)	=
>	(大於)	>
>R	(到回疊)	>R
?DUP	? 重覆	? ㄅㄨㄉ / ㄅㄨㄉ
@	(取得)	@
ABS	絕對值	ㄅㄢㄢ / ㄅㄢㄢ ㄓ
AND	聯與	ㄅ-ㄢ / ㄅ-ㄢ ㄩ
C!	(存位)	C!
C@	(取位)	C@
CMOVE	移位元	- / ㄅㄢ ㄩ ㄢ
CMOVE>	反移位	ㄅㄢ ㄢ - / ㄅㄢ
COUNT	計算	ㄅ-ㄢ ㄤ ㄢ
D+	(雙加)	D+
D<	(雙小於)	D<

Table II.
Chinese Forth-83 Standard Words

DEPTH	深度	アラ-カクハ
DNEGATE	雙負值	アバ-カクハ
DROP	丟棄	カ-スル
DUP	重複	ダブル
EXECUTE	執行	タスル
EXIT	出來	タスル
FILL	填充	タ-ル, フィル
I	(次序)	I
J	(外序)	J
MAX	極大	マックス
MIN	極小	マイン
MOD	(除餘)	モード
NEGATE	負值	カクハ
NOT	非反	ノット
OR	或者	オア
OVER	轉疊	オーバー
PICK	選疊	ピック
R>	(到回疊)	R>
R@	(複回疊)	R@
ROLL	混轉	ロール
ROT	翻轉	ローテート
SWAP	對調	スワップ
U<	(正小於)	ウル
UM*	(正混乘)	ウム
UM/MOD	(正混除餘)	ウム/モード
XOR	僅或	エクソル

Device Layer

BLOCK	區段	ブロック
BUFFER	緩衝	バッファ
CR	換行	カーリング
EMIT	放送	エミット
EXPECT	等候	エクスペクト
FLUSH	清存	フラッシュ
KEY	字鍵	キー

(Continued on next page)

SAVE-BUFFERS	存緩衝區	5x5-5x3-4xL- <u>L</u>
SPACE	空格	3xL- <u>L</u>
SPACES	空間	3xL-4-3
TYPE	印字	-5- <u>T</u>
UPDATE	標新	5- <u>T</u> -T-5

Interpreter Layer

#	(換數)	#
#>	(停換)	#>
#S	(全換)	#S
#TIB	#終入區	# <u>5xL-8x-<u</u>
.	(提取)	.
((左括弧)	(
-TRAILING	去尾	<u\ x\>\
.	(點)	.
.((點括弧)	.(
<#	(始換)	<#
>BODY	>身體	>戸- <u>5-1</u>
>IN	>入標	>8x\ 5-5
ABORT	結束	4- <u>5</u> /4x\
BASE	基數	4- <u>5</u> /4x\
BLK	表數	5- <u>5</u> /戸x\
CONVERT	變換	5-3- <u>5</u> x\
DECIMAL	十進位	戸/4- <u>5</u> x\
DEFINITIONS	添詞	5-3- <u>5</u> /
FIND	尋找	5u\ 5- <u>5</u> /
FORGET	忘記	x\ 4- <u>5</u>
FORTH	福式	5x\ 戸\
FORTH-83	福式 - 83	5x\ 戸-83
HERE	這兒	出\ 5- <u>5</u> /
HOLD	包函	5- <u>5</u> /
LOAD	譯入	-1 8x\
PAD	填補區	5-3- <u>5</u> x\<u
QUIT	終止	5xL- <u>5</u> /

SIGN	符號	ㄔㄨ／ㄅㄩ
SPAN	延申	ㄧㄢ／ㄆㄩ
TIB	終入區	ㄓㄨㄥ ㄧㄢㄩㄱㄩ
U.	(正點)	ㄩ.
WORD	詞字	ㄕ-ㄩ

Compiler Layer

+LOOP	十轉合	ㄊㄤ ㄔㄤ ㄩㄤ
,	(入位)	,
."	(示字)	.."
:	(開始定美)	:
;	(定義結束)	;
ABORT"	結束"	ㄔ-ㄔ-ㄉㄨㄤ"
ALLOT	保留	ㄕㄤ ㄉ-ㄤ
BEGIN	開始	ㄕㄤ-ㄢ
COMPILE	編碼	ㄕ-ㄢ-ㄦㄩ
CONSTANT	常數	ㄕ-ㄢ
CREATE	創造	ㄕ-ㄢ ㄭ
DO	起承	ㄕ-ㄢ ㄭ
DOES>	操作>	ㄕ-ㄢ ㄭ
ELSE	不然	ㄕㄨ ㄭ
IF	假如	ㄔ-ㄢ ㄭ
IMMEDIATE	立即	ㄕ-ㄢ ㄭ
LEAVE	離開	ㄕ-ㄢ ㄭ
LITERAL	編數	ㄕ-ㄢ ㄭ
LOOP	轉合	ㄓㄨㄥ ㄔㄤ
REPEAT	重來	ㄔㄨㄥ ㄭ
STATE	狀態	ㄓㄨㄤ ㄭ
THEN	否則	ㄔㄨㄤ ㄭ
UNTIL	直到	ㄓ-ㄢ ㄭ
VARIABLE	變數	ㄕ-ㄢ ㄭ
VOCABULARY	字彙	ㄕ ㄤ
WHILE	正當	ㄓ-ㄢ ㄭ
[(換態)	[
[']	(立即提)	[']
[COMPILE]	[編碼]	[ㄕ-ㄢ-ㄦㄩ]
]	(回態)	

End Table

cc A Driver for a Small-C Programming System

Dr. Schreiner was kind enough to provide us with two programs for use with the Small-C compiler: cc, presented here, and p, a stand-alone Small-C preprocessor. References to p, in the following discussion are to that preprocessor, which we will publish next month. — Ed.

Once you make extensive use of a programming system consisting of a compiler, assembler, and linker, you find yourself either typing a lot of commands or using the CP/M SUBMIT facility quite a bit. The latter, however, is not very flexible. It will run unconditionally whatever commands the batch file dictates; those, regardless of argument substitution, may be more or less than what you intended.

A language like C encourages separate compilation and program composition from various source files. A combination like Jim Hendrix's Small-C compiler (DDJ, December 1982) and MicroSoft's relocating assembler and linking loader makes it quite attractive to compile as little as possible during program development. When you add a separate preprocessor for Small C and attempt to eliminate intermediate files, you find yourself typing a lot of (almost) identical commands each time you want to preprocess, compile, erase, assemble, erase, link, and so on.

cc (Listing One, page 44) is a program patterned after Dennis Ritchie's cc command in the Unix* system. It accepts options and file specifications and, through CP/M's SUBMIT feature, arranges for the proper amount of preprocessing, compiling, assembly, and loading. Essentially you type what files ought to be processed to construct a program, then cc prepares a SUBMIT file and persuades CP/M to execute it.

Features

cc is based on a runtime support that passes arguments to the main program. It expects to be called as follows:

by Axel Schreiner

Axel T. Schreiner, Universität Ulm, Sektion Meth. d. Informatik, Oberer Eselsberg, 7900 Ulm (Doran) West Germany.

*Unix is a trademark of Bell Laboratories.

cc (option) . . . file . . .

You may specify options in any order. They are generally cumulative. The following options are available:

- c Compile only; do not execute the linker
- p Preprocess only
- s Preprocess and compile only
- o filename Output of the linker is "filename"

Other options are passed on to the relevant processor, mostly to the Small-C preprocessor. p accepts the following:

- d name Define a symbolic name (=value)
- e Suppress position stamps
- i drive Search for file inclusion
- u name Undefine a symbolic name

So that p can be used prior to Hendrix's compiler, the -e option is included automatically.

Files are passed on to the appropriate processors. The more general files are processed first. Intermediate files, named after the original source files, are erased once they are no longer needed; objects, however, are retained. In the absence of the -o option, the resulting load module will be named after the first of the more general source files. cc uses the following filename extensions:

- .c Preprocessor source file
- .i Compiler source file
- .mac Macro assembler source file
- .rel Linker source file or library

If an extension is not present or is unrecognizable, the file is passed only to the linker. It is thus quite simple to pass libraries. Files are passed to the linker within each source file category in the order specified; the more general source file category, however, is passed first. This rarely produces conflicts.

The options are clearly patterned after the Unix system. The main program expects to receive pointers to the various options as a vector "argv." The number of such options, including (theoretically) the program name as first option, is also passed as an integer "argc." Options must precede the filenames.

Examples

A few sample calls might illustrate just what cc does. The first call will construct the cc command itself:

B>cc cc.c

On my system I have the shortest possible names for the language processors, and I keep most tools on disk a:. Here the following commands would be issued:

```
A>b:p-e b:cc.c >b:cc.i  
A>c b:cc.i >b:cc.mac  
A>era b:cc.i  
A>m =b:cc  
A>era b:cc.mac  
A>1 b:cc/n/e,b:cc,c/s  
A>b:
```

Assume that submit.c is compiled separately.

```
B>cc -c submit.c  
A>b:p-e b:submit.c >b:submit.i  
A>c b:submit.i >b:submit.mac  
A>era b:submit.i  
A>m =b:submit  
A>era b:submit.mac  
A>b:
```

Subsequently we can compile cc.c and link it as follows:

```
B>cc submit cc.c  
A>b:p-e b:cc.c >b:cc.i  
A>c b:cc.i >b:cc.mac  
A>era b:cc.i  
A>m =b:cc  
A>era b:cc.mac  
A>1 b:cc/n/e,b:cc,submit,c/s  
A>b:
```

It should be clear why such a driver program might be useful not only for Small C.

Implementation

The system driver basically has the following structure:

- obtain and save options, initialize
- obtain, test, and save files
- for each category of source files run preprocessor, compiler, assembler as required
- run linker as required

Most of this is accomplished by main(); one subroutine for each processor handles the problem of issuing the actual commands. This arrangement makes the system easy to adapt to other processors.

File and option lists must be circular so that they can be traversed easily in input order. Push routines maintain the lists. A list element consists of a pointer to the next list element, followed by

the string that constitutes the actual value stored. The list header points to the last element entered, which in turn points to the first element, and the circular list continues on to the last element again. The list need not be marked since you always start at the list header and thus know when you have traversed the list once.

For reasons explained below, filenames must be fully specified. If an explicit drive name is not part of the file specification, `pushf()` will add the current drive name.

The `kind()` function analyzes a file specification to determine the source category based on the file extension. All "unknown" extensions are considered to be object specifications. This allows linker libraries to pass through correctly. All other files must be existing source files. To detect trivial errors early, `kind()` tests all source files for existence by opening them for reading.

submit()

The most interesting aspect of `cc` clearly is its use of the CP/M SUBMIT feature. During a warm start on drive a:, the CP/M console command processor (CCP) checks if a file `$$$.`sub exists. If it does, it is expected to contain one CCP command line (preceded by its length) per CP/M sector. The CCP takes the *last* sector from the file, truncates the file by one sector, and issues the command.

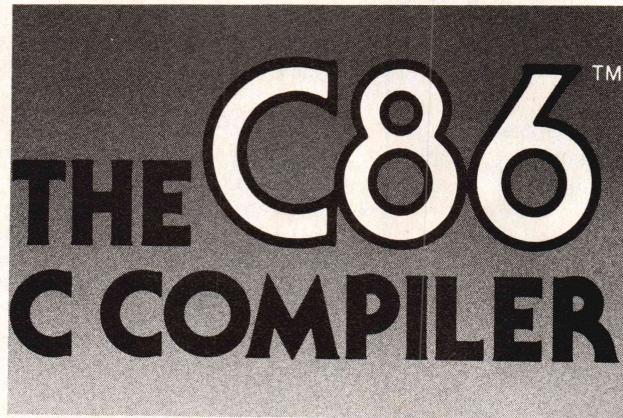
To use this feature, you need to store all command lines as a stack then write them, appropriately formatted, into the file. You also need to take care that, once the driver program exits, drive a: is selected and a warm start is performed.

The `submit()` function (Listing Two, page 52) handles all of this. If it is called with a string argument, it will stack the string (using a dynamic memory allocation function that is part of my runtime support). If it is called with a NULL argument, it will write the stack (i.e., the commands in reverse order) into the file, force the CCP to select drive a: (and user area 0) by clearing the byte at location 4, and terminate program execution through a warm start.

`submit()` issues a command to (re-)select the current drive, so that once the batch stream is completely processed the current drive is selected again. Nevertheless, during batch execution, drive a: must be selected. This is why `pushf()` fully specifies each file.

`submit()` actually appends to the end of the batch file processed by the CCP. This has a desirable effect in that `submit()` can be used from *within* a batch stream — something that is lacking in the CP/M SUBMIT utility. (It is not very difficult to position a CP/M file to end of file, especially on a sector boundary.) A simple test program (Listing Three, page 55) demonstrates this feature:

**MORE
POWER...
MORE
SPEED...**



C86, The Leading C Compiler For PC-DOS, MS-DOS Systems Is Better Than Ever.

The latest release of C86 has the features that serious programmers want the most.

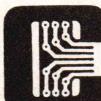
- **FAST EXECUTION** - 50% faster
- **FULL & STANDARD IMPLEMENTATION OF C** - includes all the features described by K & R. It works with the standard MS-DOS Linker and Assembler; plus programs written under UNIX can often be compiled with no changes.
- **POWERFUL OPTIONS** - include fast 8087 support; DOS2 and DOS1 support and interfaces; graphics; librarians; and much more.
- **FULL LIBRARY WITH SOURCE** - object libraries with full source code for the "large" and "small" models, software and 8087 floating point, DOS2 and DOSALL.
- **FULL RANGE OF SUPPORT PRODUCTS** - including file management, graphics, screen management, source code management, communications, and more.
- **HIGH RELIABILITY** - time proven through thousands of users.
- **DIRECT TECHNICAL SUPPORT**

**MORE POWER...MORE SPEED...
STILL ONLY \$395**

FOR MORE INFORMATION OR TO ORDER CALL:

800-922-0169

Technical Support: (201) 542-5920



COMPUTER INNOVATIONS

980 Shrewsbury Avenue, Suite DR
Tinton Falls, NJ 07724

C86 is a trademark of Computer Innovations, Inc. MS-DOS is a trademark of Microsoft. PC-DOS is a trademark of International Business Machines.

WHY FORTH ?

- Genuinely Interactive (BASIC is much less interactive)
- Encourages Modular Programs (inefficiency and cluttered syntax hamper effective modularization in compiled languages)
- Fast Execution (not even C is faster)
- Amazingly Compact Code
- Fast Program Development
- Easy Peripherals Interfacing

HS / FORTH

- Fully Optimized & Tested for: IBM-PC IBM-XT IBM-JR COMPAQ EAGLE-PC-2 TANDY 2000 LEADING EDGE and all MSDOS compatibles
- Graphics - line, rectangle, block
- Music - foreground and background
- Scaled decimal floating point
- Includes Forth-79 and Forth-83
- Full Support for DOS Files, Standard Screens and Random access DOS Screen Files
- Full Use of 8088 Instructions (not limited 8080 conversion subset of transported versions)
- Separate Segments for Code, Stack, Vocabularies, and Definition Lists - multiple sets possible
- Segment Management Support
- Full Megabyte - programs or data
- Coprocessor Support
- Multi-task, Multi-user Compatible
- Automatic Optimizer (no assembler knowledge needed)
- Full Assembler (interactive, easy to use & learn)
- Compare - BYTE Sieve Benchmark jan83 HS/FORTH 47 sec BASIC 2000 sec w/AUTOOPT 9 sec Assembler 5 sec other Forths (mostly 64k) 70-140 sec
- PS You don't have to understand this ad to love programming in HS/FORTH!

HS/FORTH with AUTO-OPT & MICRO-ASM \$220.

 Visa  Mastercard
Add \$10. shipping and handling

HARVARD SOFTWORKS

PO BOX 339
HARVARD, MA 01451
(617) 456-3021

Circle no. 28 on reader service card.

```
B>x a "b c' ""d e' f" g
a
A>b:x b c
b
A>b:x c
c
A>a:
A>b:x 'd e' f
d e
A>b:x f
f
A>a:
A>b:x g
g
A>b:
```

Quotes and double quotes can be used to hide white space within command arguments. The output shown above has been indented to show the nesting of batch streams.

Installation

Getting *cc* to run on your system might be tricky. Basically you need to decide where your processors and standard libraries reside and how they ought to be called. Also, *cc* relies on my own runtime support, which is heavily oriented towards the Unix environment.

You probably should consult Kernighan and Ritchie's book *The C Programming Language* (Prentice Hall, 1978) to learn about all the routines that are mentioned "extern" at the beginning of the program. Most of them are quite simple to construct. It is essential, however, that you provide a memory allocator, *calloc()*, that is reasonably stable. I am using a scheme where memory above the load module and below the stack is managed by a list of words, each word pointing to the next. The low bit in each such word indicates if the area past that word and up to the next one is allocated or free.

I have had access to Chapter 17 of Hendrix's book on Small C, describing his runtime support. [See this month and last month for Payne's and Hendrix's new library - Ed.] While I did not have access to the runtime support itself and therefore could not test it, I believe that installing *cc* should be quite simple. The following probably must be done:

FILE	should be defined as <i>int</i> .
_drive()	needs to access BDOS function 25.
fseek(. . . , 10)	is Hendrix's cseek(. . . , 2).
rindex()	is Hendrix's strrchr().

You can replace *_bputchar()* by storing the relevant characters in a buffer of length 128 and using Hendrix's *write()* to emit the buffer. The file pointer *_cfp* then is not needed.

I process the arguments to *main()* directly. Depending on the actual implementation, you might have to use Hendrix's function *getarg()*. I am assuming

that the storage allocator, *calloc()* / *cfree()*, supports random order release of memory.

Two runtime routines deserve special mention:

fseek(fp, 0, 10)

will position the file indicated by "fp" to the end of the last allocated CP/M sector. (In Unix the third argument for *fseek()* must be 0, 1, or 2, indicating positioning relative to the beginning of the file, the present position, and the end of the file. Additionally, I permit 8, 9, and 10, indicating sector positioning.)

_bputchar(ch)

will emit the character "ch" to the file currently indicated by "*_cfp*" without interpreting tab, return, or other characters. Since *submit()* must write CP/M sectors containing binary length information, use of this very internal routine of the runtime support is necessary. *_bputchar()* returns EOF on end of file, e.g., when the relevant disk overflows.

Deciding where your processors live and how they ought to be called is your own problem. I have a CP/M system with two 200K floppies; one of these (barely) contains Small C, the MicroSoft assembler and linker, a text editor, and my runtime library. Unless I trade the text editor for the preprocessor, I have to call the preprocessor from the second disk.

Notice that the *SUBMIT* file must be on the a: disk and that this disk must be selected while this file is processed. My processors therefore all sit on this disk, with enough room left over for the *SUBMIT* file.

Constructing the proper calls is relatively simple; if you use other systems, you may have to add a small amount of code to the program.

Part of this work was done during a sabbatical spent at the University of Illinois; in particular, the Small-C system was obtained from "UseNet."



(Listing begins on page 44)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 195.

COHERENT™ IS SUPERIOR TO UNIX* AND IT'S AVAILABLE TODAY ON THE IBM PC.

Mark Williams Company hasn't just taken a mini-computer operating system, like UNIX, and ported it to the PC. We wrote COHERENT ourselves. We were able to bring UNIX capability to the PC with the PC in mind, making it the most efficient personal computer work station available at an unbelievable price.

For the first time you get a multi-user, multitasking operating system on your IBM PC. Because COHERENT is UNIX-compatible, UNIX software will run on the PC under COHERENT.

The software system includes a C-compiler and over 100 utilities, all for \$500. Similar environments cost thousands more.

COHERENT on the IBM PC requires a hard disk and 256K memory. It's available on the IBM XT, and Tecmar, Davong and Corvus hard disks.

Available now. For additional information, call or write,

Mark Williams Company
1430 West Wrightwood, Chicago, Illinois 60614
312/472-6659



COHERENT is a trade mark of Mark Williams Company.
*UNIX is a trade mark of Bell Laboratories.

Listing One

```
/*
 *      cc - smallC compiler driver
 *      ats 5/83
 */

/*
 *      define...
 *
 *      c80      to drive Software Toolworks C80
 */

char    usage[] =
    "cc [-c|p|s] [-d n[=v]] [-e] [-i d:] [-o task] [-u n] files";

#define Csource 1      /* results of kind() */
#define Msource 2
#define Psource 3

#define Cfile    ".i"    /* for these extensions */
#define Pfile    ".c"
#define Mfile    ".mac"

#define C        "c"      /* for these processors */
#define P        "b:p"
#define M        "m"
#define L        "l"

#define CLIB     "c/s"   /* and this library search */

/*
 *      "FEATURES:"
 *
 *      If the same file name is specified with different
 *      extensions, the most general source is processed...
 *      ...and the others will fail, but be linked several times!
 */

#include <stdio.h>

/*
 *      i/o header file
 */

#define FILE    ???      type to represent files (used as FILE*)
#define stdin   ???      pre-opened standard input file
#define stdout  ???      pre-opened standard output file
#define stderr  ???      pre-opened diagnostic output file
#define NULL    0        null pointer, false
#define EOF     ???      end of file indication

/*
 */

/*
 *      special data type
 */

#define LIST     int      /* list of word or string values */

#define l_next(x)  (*(x))      /* -> next element */
#define l_str(x)   ((x)+1)      /* -> string value */
#define l_first(x) (*(x)+2)    /* -> first string value in circular list */
#define sz_STR(s)  (3+strlen(s)) /* size of string list element */
#define sz_FILE(s) (5+strlen(s)) /* size of file (+ drive) element */
```

```

/*
 *      runtime support routines
 */

extern _drive(),           /* BDOS function 25: current drive number */
      calloc(),           /* (n,l) return NULL or -> n elements of length l */
      cfree(),            /* (p) free area at p, returned by calloc() */
      exit(),              /* terminate program execution */
      fputs(),             /* (s,f) write string s on file f */
      freopen(),           /* (n,m,f) return NULL or file descriptor f,
                           closed if necessary, and reopened to read
                           (m == "r"), write ("w"), or append ("a") */
      rindex(),            /* (s,c) find c in string s end to front, return
                           NULL or ->c in s; '\0' is always found */
      strcat(),             /* (a,b) copy string b beyond string a */
      strcmp(),             /* (a,b) <, ==, > 0 as string a is <, ==, > string b */
      strcpy(),              /* (a,b) copy string b to string a */
      strlen(),             /* (s) return number of characters in string s */
      submit();             /* (s) add string s to command list; (NULL) submit */

/*
 *      global data
 */

char  buf[126],           /* to submit (length unchecked...) */
      drive[] = "?:",      /* current drive name */
      pflag,                /* run only preprocessor */
      cflag,                /* run only compiler */
      sflag;                /* run assembler */

LIST  *task,                /* task name (-o option) only */
      *popt,                /* p option list */
      *parg,                /* p argument list */
      *carg,                /* c argument list */
      *marg,                /* m argument list */
      *larg;                /* l argument list */

main(argc, argv)
    int argc;
    int *argv;
{
    char *cp;
    LIST *arg;

    /* remember current drive */
    drive[0] = _drive() + 'a';

    /* default preprocessor options */
    popt = pushs(popt, "-e");

    /* record options and flags */
    while (--argc)
    {
        cp = *++argv;
        /* options must precede files */
        if (*cp != '-')
            break;
        /* dispatch and record, accept values attached or separate */
        switch(cp[1])
        {
        case 'c':                  /* -c           run p c m */
            cflag = 1;
            break;
        case 'd':                  /* -d n[v]     p option */
        case 'i':                  /* -i d        p option */
        case 'u':                  /* -u n        p option */
            popt = pushs(popt, *argv);
            if (cp[2])
                break;
        }
    }
}

```

(Continued on next page)

Listing One

```
        if (--argc == 0)
            goto error;
        popt = pushs(popt, *++argv);
        break;
case 'e':           /* -e           p option */
        popt = pushs(popt, *argv);
        break;
case 'o':           /* -o task      name task image */
        if (cp[2])
            task = pushf(task, cp+2);
        else if (--argc == 0)
            goto error;
        else
            task = pushf(task, *++argv);
        break;
case 'p':           /* -p           run p */
        pflag = 1;
        break;
case 's':           /* -s           run p c */
        sflag = 1;
        break;
default:
        goto error;
}

/* there must be at least one file */
if (argc == 0)
{
error:    fputs(usage, stderr);
        exit();
}

/* collect files on various lists */
do
{
    switch(kind(*argv)) {
    case Psouce:
        parg = pushf(parg, *argv);
        break;
    case Csouce:
        carg = pushf(carg, *argv);
        break;
    case Msouce:
        marg = pushf(marg, *argv);
        break;
    default:
        larg = pushf(larg, *argv);
    }
    ++argv;
} while (--argc);

/* run files preprocessor -> compiler -> assembler */
if (arg = parg)
    do
    {
        arg = *arg;
        run(P, popt, l_str(arg), Pfile, Cfile);
        if (pflag)
            continue;
        run(C, NULL, l_str(arg), Cfile, Mfile);
        erase(l_str(arg), Cfile);
    }
    while (arg != NULL);
```

(Continued on page 48)

C BUILDING BLOCKS

Save a year of development.
Over 600 functions.
Main-frame libraries on a micro.

□ IBM PC* Building Blocks	\$149
All DOS functions, printer, asynchronous port, video, directories. Over 200 functions.	
□ Mathematics	\$ 99
Logarithms, trigonometric functions, square root.	
□ Communications	\$149
For Hayes Smartmodem†, xon/xoff, modem7 and xmodem.	
□ B-trees & Database Management	\$149
Variable length records, lists, duplicate keys.	
□ Advanced Building Blocks*	\$ 99
Julian Date, field input, graphic fill, pie, text windows.	
□ Archive Utility	\$ 49
Many files to one file, View/print while in archive.	

Credit Cards Accepted. Single User License.

Multiuser licenses available. (\$7.00 Handling/Mass. add 5%)

NOVUM
ORGANUM

29 Egerton Road, Arlington, MA 02174

Tel: (617) 273-4711

(TM) IBM*

(TM) Hayes Microcomputer Products†

*Require PC Building Blocks.

Circle no. 42 on reader service card.

GTEK INC. (601) 467-8048

EPROM PROGRAMMER

Compatible w/ all RS 232 serial interface port * Auto select baud rate * With or without handshaking * Bidirectional Xon/Xoff and CTS/DTR supported * Read, pin compatible ROMS * No personality modules * Intel, Motorola, MCS86, Hex formats * Split facility for 16 bit data paths * Read, program, formatted list commands * Interrupt driven, program and verify real time while sending data * Program single byte, block, or whole EPROM * Intelligent diagnostics discern bad and erasable EPROM * Verify erasure and compare commands * Busy light * Complete w/ Textool zero insertion force socket and integral 120 VAC power (240 VAC/50Hz available)

DR Utility Package allows communication with 7128, 7228, and 7956 programmers from the CP/M command line. Source Code is provided. PGX utility package allows the same thing, but will also allow you to specify a range of addresses to send to the programmer. Verify, set the EPROM type.

MODEL 7316 PAL PROGRAMMER
Programs all series 20 PALS. Software included for compiling PAL source codes.

Software Available for CPM¹, ISIS².

TRS-DOS³, MSDOS⁴.

- TM of Digital Research Corp.
- TM of Intel Corp.
- TM of Tandy Corp.
- TM of Microsoft.

Post Office Box 289
Waveland, Mississippi 39576
(601)-467-8048

DEVELOPMENT HARDWARE/SOFTWARE

HIGH PERFORMANCE/ COST RATIO

INC. (601) 467-8048

EPROM PROGRAMMER

INC. (601) 467-8

Listing One

```
        if (sflag)
            continue;
        asm(l_str(arg));
        erase(l_str(arg), Mfile);
    } while (arg != parg);
if (pflag)
    goto done;

/* run files compiler -> assembler */
if (arg = carg)
    do
    {
        arg = *arg;
        run(C, NULL, l_str(arg), Cfile, Mfile);
        if (sflag)
            continue;
        asm(l_str(arg));
        erase(l_str(arg), Mfile);
    } while (arg != carg);
if (sflag)
    goto done;

/* run files assembler */
if (arg = marg)
    do
    {
        arg = *arg;
        asm(l_str(arg));
    } while (arg != marg);
if (cflag)
    goto done;

/* run (MicroSoft) linker */
/* L task/n/e, parg..., carg..., marg..., larg..., CLIB */
/* note that we do not explicitly check the length of this command */

strcpy(buf, L);
strcat(buf, " ");

/* decide on output file name */
if (task)
    strcat(buf, l_str(task));
else if (parg)
    strcat(buf, l_first(parg));
else if (carg)
    strcat(buf, l_first(carg));
else if (marg)
    strcat(buf, l_first(marg));
else if (larg)
    strcat(buf, l_first(larg));
else
{
    strcat(buf, drive);
    strcat(buf, "task");
}
strcat(buf, "/n/e");

/* add all modules */
if (arg = parg)
    do
    {
        arg = *arg;
        strcat(buf, ",");
        strcat(buf, l_str(arg));
    } while (arg != parg);
```

```

if (arg == cargs)
    do
    {
        arg = *arg;
        strcat(buf, ",");
        strcat(buf, l_str(arg));
    } while (arg != cargs);
if (arg == margs)
    do
    {
        arg = *arg;
        strcat(buf, ",");
        strcat(buf, l_str(arg));
    } while (arg != margs);
if (arg == larg)
    do
    {
        arg = *arg;
        strcat(buf, ",");
        strcat(buf, l_str(arg));
    } while (arg != larg);

/* add smallC library and submit */
strcat(buf, ",");
strcat(buf, CLIB);
submit(buf);

/* submit the batch stream */
done:
submit(NULL);
}

/*
 *      circular list routines
 */
pushs(l, s)           /* attach string to list */
LIST *l;              /* list */
char *s;              /* string */
{
LIST *ri;             /* new element */

if ((r = calloc(sz_STR(s), 1)) == NULL)
{
    fputs("no room", stderr);
    exit();
}
strcpy(l_str(r), s);
/* empty list: link first element to itself */
if (l == NULL)
    return l_next(r) = r;
/* nonempty list: tie element into it */
l_next(r) = l_next(l);
return l_next(l) = r;
}

pushf(l, f)           /* attach file name to list */
LIST *l;              /* list */
char *f;              /* file name */
{
LIST *ri;             /* new element */

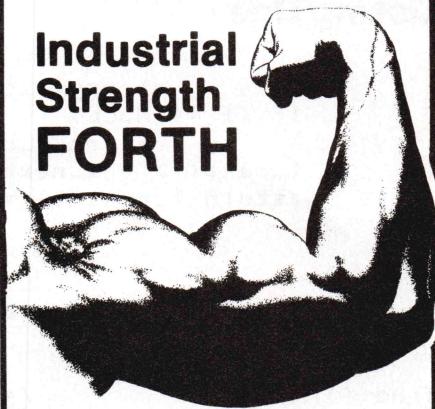
if ((r = calloc(sz_FILE(f), 1)) == NULL)
{
    fputs("no room", stderr);
    exit();
}
if (f[1] != ':')      /* use current drive */
{
    strcpy(l_str(r), drive);
    strcat(l_str(r), f);
}
else                  /* explicit drive */

```

(Continued on next page)

Multiuser/Multitasking
for 8080, Z80, 8086

Industrial Strength FORTH



TaskFORTH™

The First
Professional Quality
Full Feature FORTH
System at a micro price*

LOADS OF TIME SAVING PROFESSIONAL FEATURES:

- ★ Unlimited number of tasks
- ★ Multiple thread dictionary, superfast compilation
- ★ Novice Programmer Protection Package™
- ★ Diagnostic tools, quick and simple debugging
- ★ Starting FORTH, FORTH-79, FORTH-83 compatible
- ★ Screen and serial editor, easy program generation
- ★ Hierarchical file system with data base management

* Starter package \$250. Full package \$395. Single user and commercial licenses available.

If you are an experienced FORTH programmer, this is the one you have been waiting for! If you are a beginning FORTH programmer, this will get you started right, and quickly too!

**Available on 8 inch disk
under CP/M 2.2 or greater
also
various 5 1/4" formats
and other operating systems**

**FULLY WARRANTIED,
DOCUMENTED AND
SUPPORTED**



**DEALER
INQUIRIES
INVITED**



Shaw Laboratories, Ltd.
24301 Southland Drive, #216
Hayward, California 94545
(415) 276-5953

Circle no. 64 on reader service card. 49

Listing One

```

        strcpy(l_str(r), f);
    if (l == NULL)
        return l_next(r) = r;
    l_next(r) = l_next(l);
    return l_next(l) = r;
}

/*
 *      source file type analysis
 */

kind(f)           /* determine type of source */
{
    char *f;      /* file name */
    char *p;

    if (p = rindex(f, '.'))
        if (strcmp(p, Pfile) == 0)
        {
            if (freopen(f, "r", stdin) == NULL)
                goto badfile;
            *p = '\0';
            return Psource;
        }
        else if (strcmp(p, Cfile) == 0)
        {
            if (freopen(f, "r", stdin) == NULL)
                goto badfile;
            *p = '\0';
            return Csource;
        }
        else if (strcmp(p, Mfile) == 0)
        {
            if (freopen(f, "r", stdin) == NULL)
                goto badfile;
            *p = '\0';
            return Msource;
        }
    return 0;
badfile:
    fputs("cannot open ", stderr);
    fputs(f, stderr);
    exit();
}

/*
 *      routines to produce command calls
 */

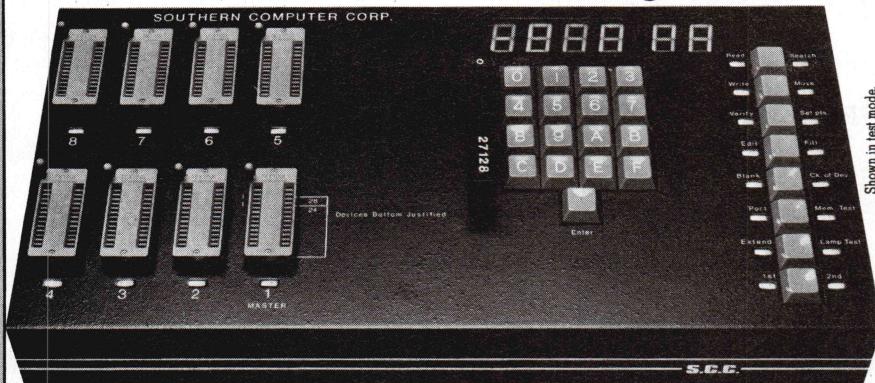
run(cmd, opt, fnm, in, out) /* run smallC task */
{
    /* cmd opt... fnm.in > fnm.out */
    char *cmd;      /* command name */
    LIST *opt;      /* option list */
    char *fnm;      /* file name to process */
    char *in;       /* input file extension */
    char *out;      /* output file extension */
    LIST *p;

    strcpy(buf, cmd);
    if (p = opt)
        do
        {
            p = l_next(p);
            strcat(buf, " ");
            strcat(buf, l_str(p));
        }
        while (p != opt);
    if (in)
        strcat(buf, in);
    if (out)
        strcat(buf, out);
    if (fnm)
        strcat(buf, fnm);
    if (cmd)
        strcat(buf, cmd);
    if (buf[0] == '\0')
        strcpy(buf, cmd);
}

```

(Continued on page 52)

The Cost Efficient EPROM Programmer!



DISPLAY Bright 1" high display system Progress indicated during programming Error messages

KEYBOARD Full travel entry keys Auto repeat Illuminated function indicators

INTERFACE RS-232C for data transfer 110-19.2K baud X-on X-off control of serial data

FUNCTIONS Fast and standard programming algorithms

Single key commands Search finds data strings up to 256 bytes long Electronic signatures for easy data error I.D. "FF" skipping for max programming speed User sets memory boundaries 15 commands including move, edit, fill, search, etc. functions Extended mode reads EPROM sets

GENERAL Stand alone operation, *external terminal not needed for full command*

set Total support 28 pin sockets Faulty EPROMS indicated at socket Programs 1 to 128K devices Built in diagnostics No calibration required No personality modules to buy Complete with 128K buffer Only

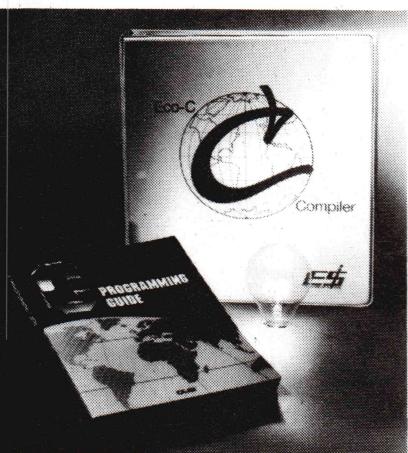
\$995.00 COMPLETE

Dealer inquiries welcome.

SOUTHERN COMPUTER CORPORATION
3720 N. Stratford Rd., Atlanta, GA 30342, 404-231-5363

Circle no. 71 on reader service card.

Lower Price!



We make C easy...

... and work!

Whether you're a seasoned pro or just getting started with C, our Eco-C C compiler has everything you need.

- A full C compiler, including long, float and double.
- A library of more than 100 functions for faster program development.

- The compiler generates assembler output (Zilog mnemonics) for processing with Microsoft's MACRO 80 assembler and linker, both of which are included in the price.
- Extremely efficient code (e.g., Knuth's "seive" executes in 15.8 seconds).
- For a limited time, we include a copy of the **C Programming Guide**. The Guide and the Eco-C compiler provide an excellent environment for learning C.

Perhaps the best news is that we've lowered the price of Eco-C to \$250.00 and it includes a user's manual, the **Guide** and MACRO 80. Eco-C requires a Z80 CPU and CP/M (an 8088 version in the 2nd quarter). To order, call or write:



6413 N. College Ave.
Indianapolis, IN 46220
(317) 255-6476

Trademarks: Eco-C (Ecosoft), MACRO 80 (Microsoft), CP/M (Digital Research), Z80 (Zilog)



Circle no. 22 on reader service card.

LISP FOR THE IBM PERSONAL COMPUTER.

THE PREMIER LANGUAGE
OF ARTIFICIAL
INTELLIGENCE FOR
YOUR IBM PC.

■ DATA TYPES

Lists and Symbols
Unlimited Precision Integers
Floating Point Numbers
Character Strings
Multidimensional Arrays
Files
Machine Language Code

■ MEMORY MANAGEMENT

Full Memory Space Supported
Dynamic Allocation
Compacting Garbage Collector

■ FUNCTION TYPES

EXPR/FEXPR/MACRO
Machine Language Primitives
Over 190 Primitive Functions

■ IO SUPPORT

Multiple Display Windows
Cursor Control
All Function Keys Supported
Read and Splice Macros
Disk Files

■ POWERFUL ERROR RECOVERY

■ 8087 SUPPORT

■ COLOR GRAPHICS

■ LISP LIBRARY

Structured Programming Macros
Editor and Formatter
Package Support
Debugging Functions
.OBJ File Loader

■ RUNS UNDER PC-DOS 1.1 or 2.0

IQLISP

5 1/4" Diskette and Manual \$175.00
Manual Only \$30.00

fq *Integral Quality*

P.O. Box 31970
Seattle, Washington 98103-0070
(206) 527-2918

Washington State residents add sales tax.
VISA and MASTERCARD accepted.
Shipping included for prepaid orders.

Circle no. 29 on reader service card.

Listing One

```
        } while (p != opt);
    strcat(buf, " ");
    strcat(buf, fnm);
#ifndef c80
    strcat(buf, in);
    strcat(buf, " >");
    strcat(buf, fnm);
    strcat(buf, out);
#endif
    submit(buf);
}

asm(fnm)           /* run (MicroSoft) macro assembler */
/* M = fnm */
char *fnm;         /* file name to process */
{
    strcpy(buf, M);
    strcat(buf, " =");
    strcat(buf, fnm);
    submit(buf);
}

erase(fnm,ext)     /* erase intermediate file */
/* era fnm.ext */
char *fnm;         /* file name */
char *ext;          /* extension */
{
    strcpy(buf, "era ");
    strcat(buf, fnm);
    strcat(buf, ext);
    submit(buf);
}
```

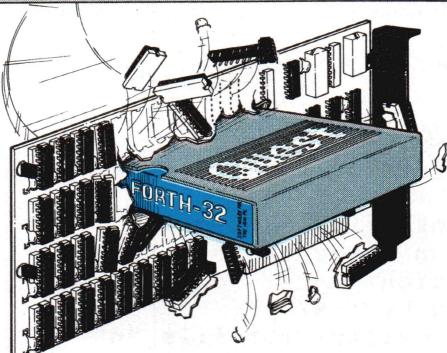
End Listing One

Listing Two

```
/*
*      submit() -- submit commands to CP/M batch
*      ats 5/83
*/
#define DSKBYTE 4           /* BDOS/CCP selected disk, user# */

/*
*      needed from the i/o header file
#define FILE    ???      type to represent files
#define NULL    0        null pointer, false
#define EOF     ???      end of file indication
*/
/*
*      runtime support routines
*/
```

(Continued on page 54)



Break Through the 64K Barrier!

FORTH-32™ lets you use up to one megabyte of memory for programming. A Complete Development System! Fully Compatible Software and 8087 Floating Point Extensions.



303 Williams Ave.
Huntsville, AL 35801
(205) 533-9405

Call today toll-free or contact a participating Computerland store.

800-558-8088

Now available for the IBM PC, PC-XT, COMPAQ, COLUMBIA MPC, and other PC compatibles!

IBM, COMPAQ, MPC, and FORTH-32 are trademarks of IBM, COMPAQ, Columbia Data Products, and Quest Research, respectively.

Circle no. 55 on reader service card.

MU FORTH
FOR YOUR IBM PC® ...

Now the advantages of Forth are available for your IBM PC®. Easy to use for both experienced programmers and beginners too, Forth offers the advantages of speed, flexibility, and ease of use.

The MU Forth Package Provides:

- Source and object code user documentation
- Commented source listings in IBM and macro assembler format
- Turtle graphics, sound, and music support
- Keyboard and video support sample programs

Requires:

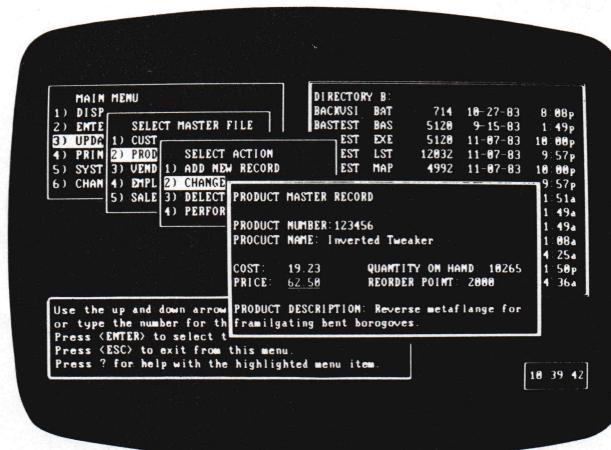
Your IBM PC, 16K memory, 1 disk drive, PC DOS 1.X, color or monochrome, 40 or 80 column display.

\$75
The Price

Available from: Art Arizpe
MU Software
115 Cannongate III, Nashua, NH 03063
603-880-9416
SEND CHECK OR MONEY ORDER

Circle no. 40 on reader service card.

WE DO WINDOWS.



Don't just put your applications in windows—put windows in your applications with VSI—the window manager.

VSI is a high-speed screen management tool. You can create up to 255 simultaneously active overlapping windows—large or small—for any application program. Read to or write from any window and display them with borders and user declared priorities. VSI is callable from any compiled language and supports all color and monochrome video attributes.

Cut Development Time

VSI's powerful primitives simplify your screen management chores with a complete library of functions. And you can preview and edit your screen layout before you actually program it.

But that's only the beginning.

Free Demo Disk

Our free hands-on demo disk will have you doing windows, too. Return the coupon with \$4.50 for postage and handling.

MasterCard or Visa accepted with phone orders only.

VSI is used with IBM PC, XT and compatibles as well as TI Professional, and Wang PC.

I develop software for 8086/8088 based machines and I want to do windows, too. I'm enclosing \$4.50 for postage and handling. Please send me your free demo disk. My business card is attached. (Offer expires December 31, 1984)

Computer: _____

Name: _____

Company: _____

Address: _____

City: _____ State: _____ Zip: _____



Copyright 1984 Amber Systems, Inc.

AMBER SYSTEMS, INC.
1171 S. Sunnyvale-Saratoga Road
San Jose CA 95129
(408) 996-1883

Circle no. 2 on reader service card.

Listing Two

```
extern _bputchar(), /* (c) write char c uninterpreted to FILE *_cfp */
      _cfp,
      _drive(),
      calloc(),
      exit(),
      fclose(),
      fopen(),
      fseek(),
      strcpy(),
      strlen();

/*
 *      global data
 */

int *_submit;          /* stack of submitted buffers */

submit(s)              /* add command to CP/M job stream */
                      /* returns argument or NULL */
                      /* on NULL argument: submits and exits */
{
    char * si;          /* command to submit */
    FILE *fp;           /* need to use block i/o, NOT putc */
    char *cp;
    int *wp, i;

    /* if string argument, save it as a command */
    if (s)
    {
        if (strlen(s) > 126)
            return NULL; /* too long */
        if ((wp = calloc(strlen(s)+2+2,1)) == NULL)
            return NULL; /* no room */
        *wp = _submit;
        _submit = wp;
        cp = _submit+1;

        *cp = strlen(s); /* length byte */
        strcpy(cp+1, s); /* text */
        return s;
    }

    /* if anything to submit - write it to a:$$$.sub */
    if (_submit)
    {
        if ((fp = fopen("a:$$$.sub", "a")) == NULL)
            return NULL; /* cannot make batch file */
        if (fseek(fp, 0, 10) == -1)
            goto error;
        _cfp = fp; /* for _bputchar() */

        /* last command: reselect current drive */
        _bputchar(2);
        _bputchar(_drive()+'a');
        _bputchar(':');
        for (i = 3; i < 127; ++i)
            _bputchar(0);
        /* check last byte in sector for overflow */
        if (_bputchar(0))
            goto error;

        /* other commands from stack */
        do

```

```

        {
            cp = _submit+1;
            for (i = 0; i < 127; ++i)
            {
                _bputchar(*cp);
                if (*cp)
                    ++cp;
            }
            if (_bputchar(*cp) == EOF)
            {
                fclose(fp);
                return NULL; /* overflow */
            }
        } while (_submit = *_submit);

        /* signal CCP to select a: and user 0 */
        cp = DSKBYTE;
        *cp = 0;
    }

    /* exit this program and perform warm start */
    exit();
}

```

End Listing Two

Listing Three

```

#include <stdio.h>
extern strcpy(), strcat(), submit(), puts();

main(argc, argv)
    int argc;
    int *argv;
{
    int i;
    char buf[128];

    switch (argc) {
    default:
        strcpy(buf, "b:x ");
        for (i=2; i<argc; ++i)
        {
            strcpy(buf+4, argv[i]);
            submit(buf);
        }
    case 2:
        puts(argv[1]);
    case 1:
        ;
    }
    if (submit(NULL) == NULL)
        puts("oops");
}

```

End Listing Three

A New Library for Small-C [Part II]

In December 1982 and January 1983 we presented version 2.0 of the Small-C compiler by James Hendrix. Last month Hendrix and Ernest Payne presented their enhanced library for the compiler, as well as code which upgrades the compiler to version 2.1. This month completes the listing of the library. —Ed.

Small-C Library Listing Two

(Continued from May issue page 81)

File: FSCANF.C

```
1 #define NOCCARGC /* no argument count passing */
2 /*
3 ** Yes, that is correct. Although these functions use an
4 ** argument count, they do not call functions which need one.
5 */
6 #include stdio.h
7 /*
8 ** fscanf(fd, ctlstring, arg, arg, ...) - Formatted read.
9 ** Operates as described by Kernighan & Ritchie.
10 ** b, c, d, o, s, u, and x specifications are supported.
11 ** Note: b (binary) is a non-standard extension.
12 */
13 fscanf(argc int argc;
14 int *nxtarg;
15 nxtarg = CCARGC() + &argc;
16 return (_scan(*(--nxtarg), --nxtarg));
17 )
18 /*
19 **
20 ** scanf(ctlstring, arg, arg, ...) - Formatted read.
21 ** Operates as described by Kernighan & Ritchie.
22 ** b, c, d, o, s, u, and x specifications are supported.
23 ** Note: b (binary) is a non-standard extension.
24 */
25 scanf(argc int argc;
26 return (_scan(stdin, CCARGC() + &argc - 1));
27 )
28 /*
29 **
30 ** _scan(fd, ctlstring, arg, arg, ...) - Formatted read.
```

by James Hendrix and Ernest Payne

Copyright © 1984 L. E. Payne and J. E. Hendrix.

James Hendrix, Box 8378, University, MS 38677-8378.
Ernest Payne, 1331 W. Whispering Hills Dr., Tucson, AZ 85704.

Unix is a trademark of AT&T Bell Labs. MACRO-80 is a trademark of Microsoft Inc.

```
31 ** Called by fscanf() and scanf().
32 */
33 _scan(fd,nxtarg) int fd, *nxtarg; {
34     char *carg, *ctl, *unsigned;
35     int *narg, wast, ac, width, ch, cnv, base, ovfl, sign;
36     ac = 0;
37     ctl = *nxtarg--;
38     while(*ctl) {
39         if(ispace(*ctl)) (++ctl; continue);
40         if(*ctl++ != '%') continue;
41         if(*ctl == '*') {narg = carg = &wast; ++ctl;}
42         else
43             narg = carg = *nxtarg--;
44         ctl += utoi(ctl, &width);
45         if(!width) width = 32767;
46         if(!(cnv = *ctl++)) break;
47         while(ispace(ch = fgetc(fd))) ;
48         if(ch == EOF) {if(ac) break; else return(EOF);}
49         ungetc(ch,fd);
50         switch(cnv) {
51             case 'c':
52                 *carg = fgetc(fd);
53                 break;
54             case 's':
55                 while(width--) {
56                     if(*carg = fgetc(fd)) == EOF) break;
57                     if(ispace(*carg)) break;
58                     if(carg != &wast) ++carg;
59                 }
60                 *carg = 0;
61                 break;
62             default:
63                 switch(cnv) {
64                     case 'b': base = 2; sign = 1; ovfl = 32767; break;
65                     case 'd': base = 10; sign = 0; ovfl = 3276; break;
66                     case 'o': base = 8; sign = 1; ovfl = 8191; break;
67                     case 'u': base = 10; sign = 1; ovfl = 6553; break;
68                     case 'x': base = 16; sign = 1; ovfl = 4095; break;
69                     default: return (ac);
70                 }
71                 *narg = unsigned = 0;
72                 while(width-- && !ispace(ch=fgetc(fd)) && ch!=EOF) {
73                     if(!sign)
74                         if(ch == '-') {sign = -1; continue;}
75                         else sign = 1;
76                     if(ch < '0') return (ac);
77                     if(ch >= 'a') ch -= 87;
78                     else if(ch >= 'A') ch -= 55;
79                     else ch -= '0';
80                     if(ch >= base || unsigned > ovfl) return (ac);
81                     unsigned = unsigned * base + ch;
82                 }
83                 *narg = sign * unsigned;
84             }
85             ++ac;
86         }
87     }
88 }
```

```
86  return (ac);
87 }
88
```

File: FWRITE.C

```
1 #define NOCCARGC /* no argument count passing */
2 #include clib.def
3 extern int _status[];
4 /*
5 ** Item-stream write to fd.
6 ** Entry: buf = address of source buffer
7 **         sz = size of items in bytes
8 **         n = number of items to write
9 **         fd = file descriptor
10 ** Returns a count of the items actually written or
11 ** zero if an error occurred.
12 ** May use perror(), as always, to detect errors.
13 */
14 fwrite(buf, sz, n, fd) char *buf; int sz, n, fd;
15 int cnt;
16 if((cnt = write(fd, buf, n*sz)) == -1) return (0);
17 return (cnt);
18 }
19
20 /*
21 ** Binary-stream write to fd.
22 ** Entry: fd = file descriptor
23 **         buf = address of source buffer
24 **         n = number of bytes to write
25 ** Returns a count of the bytes actually written or
26 ** -1 if an error occurred.
27 ** May use perror(), as always, to detect errors.
28 */
29 write(fd, buf, n) int fd, n; char *buf;
30 char *cnt; /* fake unsigned */
31 cnt = 0;
32 while(n--) {
33     _write(*buf++, fd);
34     if(_status[fd] & ERRBIT) return (-1);
35     ++cnt;
36 }
37 return (cnt);
38 }
```

File: GETARG.C

```
1 #define NOCCARGC /* no argument count passing */
2 #include stdio.h
3 /*
4 ** Get command line argument.
5 ** Entry: n    = Number of the argument.
6 **         s    = Destination string pointer.
7 **         size = Size of destination string.
8 **         argc = Argument count from main().
9 **         argv = Argument vector(s) from main().
10 ** Returns number of characters moved on success,
11 ** else EOF.
12 */
13 getarg(n,s,size,argc,argv)
14 int n; char *s; int size, argc, argv[];
15 char *str;
```

```
16 int i;
17 if(n < 0 || n >= argc) {
18     *s = NULL;
19     return EOF;
20 }
21 i = 0;
22 str=argv[n];
23 while(i<size) {
24     if((s[i]==str[i])==NULL) break;
25     ++i;
26 }
27 s[i]=NULL;
28 return i;
29 }
```

File: GETCHAR.C

```
1 #define NOCCARGC /* no argument count passing */
2 #include stdio.h
3 /*
4 ** Get next character from standard input.
5 */
6 getchar() {
7     return (fgetc(stdin));
8 }
```

File: ISALNUM.C

```
1 /*
2 ** return 'true' if c is alphanumeric
3 */
4 isalnum(c) int c;
5 return ((c<='z' && c>='a') ||
6         (c<='Z' && c>='A') ||
7         (c<='9' && c>='0'));
8 }
```

File: ISALPHA.C

```
1 /*
2 ** return 'true' if c is alphabetic
3 */
4 isalpha(c) int c;
5 return ((c<='z' && c>='a') || (c<='Z' && c>='A'));
6 }
```

File: ISASCII.C

```
1 /*
2 ** return 'true' if c is an ASCII character (0-127)
3 */
4 isascii(c) char *c;
5 /* c is a simulated unsigned integer */
6 return (c <= 127);
7 }
```

File: ISATTY.C

```
1 extern int _device[];
2 /*
3 ** Return "true" if fd is a device, else "false"
4 */
```

(Continued on next page)

Small-C Library (Listing Continued)

Listing Two

```
4 */
5 isatty(fd) int fd; {
6   return (_device[fd]);
7 }
```

File: ISCNTRL.C

```
1 /*
2 ** return 'true' if c is a control character
3 ** (0-31 or 127)
4 */
5 iscntrl(c) char *c; {
6   /* c is a simulated unsigned integer */
7   return ((c <= 31) || (c == 127));
8 }
```

File: ISCONS.C

```
1 #include stdio.h
2 #include clib.def
3 extern int _device[];
4 /*
5 ** Determine if fd is the console.
6 */
7 iscons(fd) int fd; {
8   return (_device[fd] == CPMCON);
9 }
```

File: ISDIGIT.C

```
1 /*
2 ** return 'true' if c is a decimal digit
3 */
4 isdigit(c) int c; {
5   return (c<='9' && c>='0');
6 }
```

File: ISGRAPH.C

```
1 /*
2 ** return 'true' if c is a graphic character
3 ** (33-126)
4 */
5 isgraph(c) int c; {
6   return (c>=33 && c<=126);
7 }
```

File: ISLOWER.C

```
1 /*
2 ** return 'true' if c is lower-case alphabetic
3 */
```

```
3 */
4 islower(c) int c; {
5   return (c<='z' && c>='a');
6 }
```

File: ISPRINT.C

```
1 /*
2 ** return 'true' if c is a printable character
3 ** (32-126)
4 */
5 isprint(c) int c; {
6   return (c>=32 && c<=126);
7 }
```

File: ISPUNCT.C

```
1 #define NOCCARGC /* no argument count passing */
2 /*
3 ** return 'true' if c is a punctuation character
4 ** (all but control and alphanumeric)
5 */
6 ispunct(c) int c; {
7   return (!isalnum(c) && !iscntrl(c));
8 }
```

File: ISSPACE.C

```
1 /*
2 ** return 'true' if c is a white-space character
3 */
4 isspace(c) int c; {
5   /* first check gives quick exit in most cases */
6   return(c<=' ' && (c>=' ' || (c<=13 && c>=9)));
7 }
8
```

File: ISUPPER.C

```
1 /*
2 ** return 'true' if c is upper-case alphabetic
3 */
4 isupper(c) int c; {
5   return (c<='Z' && c>='A');
6 }
```

File: ISXDIGIT.C

```
1 /*
2 ** return 'true' if c is a hexadecimal digit
3 ** (0-9, A-F, or a-f)
```

(Continued on page 60)

LIFEBOAT™ Associates:
The full support software
source.

Reach for the programming horizon of the 80's with Lattice™ C, the fastest C compiler.

Set the course of your next software project towards greater power and portability by selecting Lattice C, recognized as the finest and fastest 16-bit C compiler. Lattice C, the full implementation of Kernighan and Ritchie, continues to lead the way by announcing full memory management up to 1 Megabyte. Major software houses including Microsoft, MicroPro™ and Sorcim™ are using Lattice C to develop their programs.

Lifeboat offers Lattice C with a tested set of software tools to provide a complete development system including:

HALO™
PANEL™
PMATE™
PLINK™-86
C-FOOD SMORGASBORD™
LATTICE WINDOW™
FLOAT87™

Color graphic primitives
Screen design aid
Customizable program editor
Overlay linkage editor
Screen and I/O utilities
Multi-window functions
8087 floating point math

Lattice C is available for a wide variety of 16-bit personal computers including IBM®, NCR®, Texas Instruments, Victor, Wang and other microcomputers running PC™-DOS, MS™-DOS and CP/M86™.

Call LIFEBOAT at 212-860-0300 for free information on the Lattice C family of software development tools.

LIFEBOAT
Associates

LIFEBOAT
Associates

1651 Third Avenue
New York, NY 10028
212-860-0300

Please send me free information on: Name

- Lattice and development tools
- Corporate purchase program
- Dealer program
- OEM agreements

- Send me the complete LIFEBOAT software catalog. \$3.00 enclosed for postage and handling.

LATTICE, C-FOOD SMORGASBORD and
LATTICE WINDOW,™ Lattice, Inc.
MICROPRO,™ International Corp.
SORCIM,™ Sorcim Corp.

LIFEBOAT,™ Intersoft Corp.
HALO,™ Media Cybernetics
PANEL,™ Roundhill Computer, Ltd.
PMATE and PLINK,™ Phoenix Software

FLOAT87,™ Microfloat
IBM and PC,™ International Business Machines
MS,™ Microsoft
CP/M86,™ Digital Research

Small-C Library (Listing Continued)

Listing Two

```
4 */
5 isxdigit(c) int c; {
6     return ((c<='f' && c>='a') ||
7             (c<='F' && c>='A') ||
8             (c<='9' && c>='0'));
9 }
```

File: ITOA.C

```
1 #define NOCCARGC /* no argument count passing */
2 /*
3 ** itoa(n,s) - Convert n to characters in s
4 */
5 itoa(n, s) char *s; int n; {
6     int sign;
7     char *ptr;
8     ptr = s;
9     if ((sign = n) < 0) /* record sign */
10    n = -n; /* make n positive */
11    do { /* generate digits in reverse order */
12        *ptr++ = n % 10 + '0'; /* get next digit */
13    } while ((n = n / 10) > 0); /* delete it */
14    if (sign < 0) *ptr++ = '-';
15    *ptr = '\0';
16    reverse(s);
17 }
```

File: ITOAB.C

```
1 #define NOCCARGC /* no argument count passing */
2 /*
3 ** itoab(n,s,b) - Convert "unsigned" n to characters in s
4 ** using base b.
5 **          NOTE: This is a non-standard function.
6 */
7 itoab(n, s, b) int n; char *s; int b; {
8     char *ptr;
9     int lowbit;
10    ptr = s;
11    do {
12        lowbit = n & 1;
13        n = (n >> 1) & 32767;
14        *ptr = ((n % b) << 1) + lowbit;
15        if(*ptr < 10) *ptr += '0'; else *ptr += 55;
16        ++ptr;
17    } while(n != 0);
18    *ptr = 0;
19    reverse(s);
20 }
21
```

File: ITOD.C

```
1 #include stdio.h
```

```
2 /*
3 ** itod -- convert nbr to signed decimal string of width sz
4 **          right adjusted, blank filled; returns str
5 */
6 **          if sz > 0 terminate with null byte
7 **          if sz = 0 find end of string
8 **          if sz < 0 use last byte for data
9 */
10 itod(nbr, str, sz) int nbr; char str[]; int sz; {
11     char sgn;
12     if(nbr<0) {nbr = -nbr; sgn='-'};
13     else sgn=' ';
14     if(sz>0) str[--sz]=NULL;
15     else if(sz<0) sz = -sz;
16     else while(str[sz]!=NULL) ++sz;
17     while(sz) {
18         str[--sz]=(nbr%10+'0');
19         if((nbr=nbr/10)==0) break;
20     }
21     if(sz) str[--sz]=sgn;
22     while(sz>0) str[--sz]=' ';
23     return str;
24 }
```

File: ITOO.C

```
1 /*
2 ** itoo -- converts nbr to octal string of length sz
3 **          right adjusted and blank filled, returns str
4 */
5 **          if sz > 0 terminate with null byte
6 **          if sz = 0 find end of string
7 **          if sz < 0 use last byte for data
8 */
9 itoo(nbr, str, sz) int nbr; char str[]; int sz; {
10    int digit;
11    if(sz>0) str[--sz]=0;
12    else if(sz<0) sz = -sz;
13    else while(str[sz]!=0) ++sz;
14    while(sz) {
15        digit=nbr&7; nbr=(nbr>>3)&8191;
16        str[--sz]=digit+48;
17        if(nbr==0) break;
18    }
19    while(sz) str[--sz]=' ';
20    return str;
21 }
```

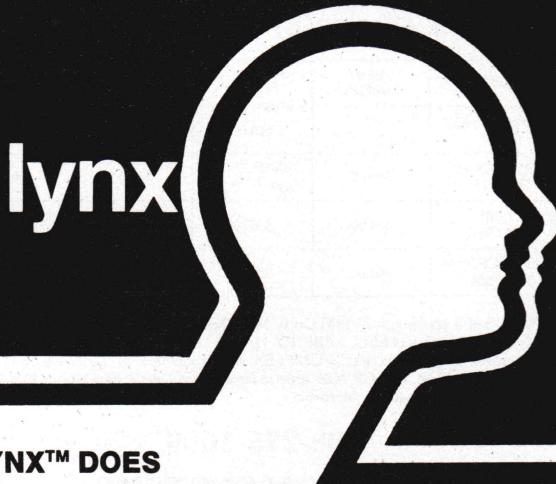
File: ITOU.C

```
1 #include stdio.h
2 /*
3 ** itou -- convert nbr to unsigned decimal string of width sz
```

(Continued on page 62)

LYNX

the professional's replacement
for Microsoft L80



LYNX™ DOES

EVERYTHING L80™ DOES AND MORE!

LYNX™ LETS YOU USE ALL AVAILABLE MEMORY.

You can create COM files that are 10K larger—without overlays—by replacing L80 with LYNX.

LYNX™ IS A FULL OVERLAY LINKER

Running twice as fast as its nearest competitor, LYNX is tree structured, multi-segmented and multi-leveled, with automatic or explicit overlay invocation. You can run programs that are larger than available memory.

LYNX™ HAS BEEN HELPING MICROSOFT FORTRAN PROGRAMMERS FOR YEARS—NOW IT IS ALSO AVAILABLE FOR MICROSOFT BASIC AND AZTEC C.™

LYNX™ IS A QUALITY PRODUCT from the same company who offers you:

- **GraTalk**, the business graphics package for Micros
- **GRAPHICS** development tools
- **2780/3780, 3270, X.25** communications
- **MICRO TO MICRO** communications



REDDING GROUP INC.

2730 High Ridge Road
Stamford, CT 06903
(203) 329-8874/Telex. 643351

\$250

LYNX and GraTalk are trademarks of Redding Group, Inc. L80 is a trademark of Microsoft. AZTEC C is a trademark of MANX Software Systems.

Small-C Library (Listing Continued)

Listing Two

```
4 **      right adjusted, blank filled; returns str
5 **
6 **      if sz > 0 terminate with null byte
7 **      if sz = 0 find end of string
8 **      if sz < 0 use last byte for data
9 */
10 itou(nbr, str, sz) int nbr; char str[]; int sz; {
11     int lowbit;
12     if(sz>0) str[--sz]=NULL;
13     else if(sz<0) sz = -sz;
14     else while(str[sz]!=NULL) ++sz;
15     while(sz) {
16         lowbit=nbr&1;
17         nbr=(nbr>>1)&32767; /* divide by 2 */
18         str[--sz]=((nbr%5)<<1)+lowbit+'0';
19         if((nbr=nbr/5)==0) break;
20     }
21     while(sz) str[--sz]=' ';
22     return str;
23 }
```

File: ITDX.C

```
1 /*
2 ** itox -- converts nbr to hex string of length sz
3 **      right adjusted and blank filled, returns str
4 **
5 **      if sz > 0 terminate with null byte
6 **      if sz = 0 find end of string
7 **      if sz < 0 use last byte for data
8 */
9 itox(nbr, str, sz) int nbr; char str[]; int sz; {
10     int digit, offset;
11     if(sz>0) str[--sz]=0;
12     else if(sz<0) sz = -sz;
13     else while(str[sz]!=0) ++sz;
14     while(sz) {
15         digit=nbr&15; nbr=(nbr>>4)&4095;
16         if(digit<10) offset=48; else offset=55;
17         str[--sz]=digit+offset;
18         if(nbr==0) break;
19     }
20     while(sz) str[--sz]=' ';
21     return str;
22 }
```

File: LEFT.C

```
1 /*
2 ** left -- left adjust and null terminate a string
3 */
4 left(str) char *str; {
5     char *str2;
6     str2=str;
7     while(*str2==' ') ++str2;
8     while(*str++ = *str2++);
9 }
```

File: LEXCMP.C

```
1 #define NOCCARGC /* no argument count passing */
2 /*
3 ** lexcmp(s, t) - Return a number <0, 0, or>0
4 **           as s is <, =, or > t.
5 */
6 lexcmp(s, t) char *s, *t;
7   while(*s == *t) {
8     if(*s == 0) return (0);
9     ++s; ++t;
10   }
11   return (lexorder(*s, *t));
12 }
13
14 */
15 ** lexorder(c1, c2)
16 **
17 ** Return a negative, zero, or positive number if
18 ** c1 is less than, equal to, or greater than c2,
19 ** based on a lexicographical (dictionary order)
20 ** collating sequence.
21 **
22 */
23 char _lex[128] = {
24   /**** NUL - / ****/
25   000,001,002,003,004,005,006,007,008,009,
26   010,011,012,013,014,015,016,017,018,019,
27   020,021,022,023,024,025,026,027,028,029,
28   030,031,032,033,034,035,036,037,038,039,
29   040,041,042,043,044,045,046,047,
30   /**** 0-9 ****/
31   065,066,067,068,069,070,071,072,073,074,
32   /**** : ; < = > ? @ ****/
33   048,049,050,051,052,053,054,
34   /**** A-Z ****/
35   075,076,077,078,079,080,081,082,083,084,085,086,087,
36   088,089,090,091,092,093,094,095,096,097,098,099,100,
37   /**** [ \ ] ^ _ ****/
38   055,056,057,058,059,060,
39   /**** a-z ****/
40   075,076,077,078,079,080,081,082,083,084,085,086,087,
41   088,089,090,091,092,093,094,095,096,097,098,099,100,
42   /**** { | } ~ ****/
43   061,062,063,064,
44   /**** DEL ****/
45   101
46   };
47
48 lexorder(c1, c2) char c1, c2;
49   return(_lex[c1] - _lex[c2]);
50 }
```

File: MALLOC.C

```
1 #define NOCCARGC /* no argument count passing */
2 #include stdio.h
3 /*
4 ** Memory allocation of size bytes.
5 ** size = Size of the block in bytes.
6 ** Returns the address of the allocated block,
7 ** else NULL for failure.
```

NGS FORTH

A FAST FORTH
OPTIMIZED FOR THE IBM
PERSONAL COMPUTER
AND MSDOS COMPATIBLES.

*79 STANDARD

*FIG LOOKALIKE MODE

*PC-DOS COMPATIBLE

*ON-LINE CONFIGURABLE

*ENVIRONMENT SAVE
& LOAD

*MULTI-SEGMENTED

*EXTENDED ADDRESSING

*AUTO LOAD SCREEN BOOT

*LINE AND SCREEN EDITORS

*DECOMPILER &
DEBUGGING AIDS

*8088 ASSEMBLER

*BASIC GRAPHICS & SOUND

*NGS ENHANCEMENTS

*DETAILED MANUAL

*INEXPENSIVE UPGRADES

*NGS USER NEWSLETTER

A COMPLETE FORTH
DEVELOPMENT SYSTEM.

PRICE: \$70

PLEASE INCLUDE \$2 POSTAGE &
HANDLING WITH EACH ORDER.
CALIFORNIA RESIDENTS:
INCLUDE 6.5% SALES TAX.



NEXT GENERATION SYSTEMS
P.O.BOX 2987
SANTA CLARA, CA. 95055
(408) 241-5909

(Continued on next page)

Circle no. 41 on reader service card.

Small-C Library (Listing Continued)

Listing Two

```
8 */
9 malloc(size) char *size; {
10  return (_alloc(size, NO));
11 }
```

File: OTOI.C

```
1 #include stdio.h
2 /*
3 ** atoi -- convert unsigned octal string to integer nbr
4 **           returns field size, else ERR on error
5 */
6 atoi(octstr, nbr) char *octstr; int *nbr; {
7  int d,t; d=0;
8  *nbr=0;
9  while((*octstr>='0')&(*octstr<='7')) {
10   t=*nbr;
11   t=(t<(3) + (*octstr++ - '0'));
12   if ((t>=0)&(*nbr<0)) return ERR;
13   d++; *nbr=t;
14  }
15  return d;
16 }
```

File: PAD.C

```
1 #define NOCCARGC /* no argument count passing */
2 /*
3 ** Place n occurrences of ch at dest.
4 */
5 pad(dest, ch, n) char *dest, *n; int ch; {
6  /* n is a fake unsigned integer */
7  while(n--) *dest++ = ch;
8 }
```

File: POLL.C

```
1 #define NOCCARGC /* no argument count passing */
2 #include stdio.h
3 #include clib.def
4 /*
5 ** Poll for console input or interruption
6 */
7 poll(pause) int pause; {
8  int i;
9  i = _bdos(DCONIO, 255);
10 if(pause) {
11  if(i == PAUSE) {
12   while(!(i = _bdos(DCONIO, 255)));
13   if(i == ABORT) exit(0);
14  return (0);
15  }
16  if(i == ABORT) exit(0);
17 }
18 return (i);
19 }
```

File: PUTCHAR.C

```
1 #define NOCCARGC /* no argument count passing */
2 #include stdio.h
3 /*
4 ** Write character to standard output.
5 */
6 putchar(ch) int ch; {
7  return (fputc(ch, stdout));
8 }
```

File: PUTS.C

```
1 #define NOCCARGC /* no argument count passing */
2 #include stdio.h
3 /*
4 ** Write string to standard output.
5 */
6 puts(string) char *string; {
7  fputs(string, stdout);
8  fputc('\n', stdout);
9 }
```

File: RENAME.C

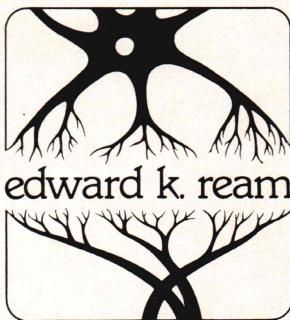
```
1 #define NOCCARGC /* no argument count passing */
2 #include stdio.h
3 #include clib.def
4 /*
5 ** Rename a file.
6 ** from = address of old filename.
7 ** to = address of new filename.
8 ** Returns NULL on success, else ERR.
9 */
10 rename(from, to) char *from, *to; {
11  char fcb[FCBSIZE];
12  pad(fcb, NULL, FCBSIZE);
13  if(!_newfcb(to, fcb) || _bdos(OPENFIL, fcb) != 255) {
14   _bdos(CLOFIL, fcb);
15   return (ERR);
16  }
17  if(_newfcb(from, fcb) &&
18   _newfcb(to, fcb+NAMEOFF2) &&
19   _bdos(RENAMF, fcb) != 255)
20  return (NULL);
21  return (ERR);
22 }
```

File: REVERSE.C

```
1 #define NOCCARGC /* no argument count passing */
2 /*
3 ** reverse string in place
4 */
5 reverse(s) char *s; {
```

(Continued on page 66)

RED



FULL SCREEN EDITOR with FULL SOURCE CODE in C for CP/M 68K or CP/M 80

- RED is a powerful yet simple full screen text editor for both programmers and writers.
- RED features 17 commands including block move, block copy, search and substitute.
- RED comes with *full* source code in standard C. RED works as is with the BDS C, Aztec CII and Digital Research Compilers.
- RED supports *all* features of your terminal. You tailor RED to your terminal with an easy-to-use configuration program.
- RED handles files as large as your disk.
- RED is guaranteed. If for any reason you are not satisfied with RED, your money will be refunded promptly.

Price: \$95.

Call today for more valuable information:
(608) 231-2952

To order, send a check or money order to:
Edward K. Ream
1850 Summit Avenue
Madison, Wisconsin 53705

Your order will be mailed to you within one week. Sorry, I can not handle credit cards. Please do not send purchase orders unless a check is included. RED is distributed only on 8 inch CP/M format disks, but other disk formats are available through third parties.

Dealer inquiries invited.

'C' COMPILER

AN OUTSTANDING VALUE

"We bought and evaluated over \$1500.00 worth of 'C' compilers... C/80 is the one we use."

Dr. Bruce E. Wampler, Aspen Software
author of "Grammatik"®

C/80 TM Full featured C Compiler for CP/M® with I/O redirection, command expansion, execution trace and profile, initializers, Macro-80 compatibility, ROMable code.

49.95

C/80 FLOATS & LONGS Adds 32 bit data types to C/80 3.0 compiler. Includes I/O and transcendental function library.

29.95

FREE CATALOG Call or write for 16 page booklet detailing our programming languages LISP/80, RATFOR, Assemblers, and 25 other CP/M products.



15233 Ventura Blvd., #1118
Sherman Oaks, CA 91403

(213) 986-4885

Dealer inquiries invited.

CP/M is a registered trademark of Digital Research Inc.

Circle no. 70 on reader service card.

FOR \$29.95, DISK INSPECTOR MAKES YOU A SHERLOCK HOLMES OF THE COMPUTER!

Ever wonder what happened to that erased or lost file? Did text suddenly disappear and can't be found? Did a bad sector do strange things to your files? Then track them down with *Disk Inspector*! Rated "Excellent" by INFOWORLD, *Disk Inspector* is a utility that pays for itself with the first recovery. Even more, *Disk Inspector* allows you to use the Auto-Load feature of CP/M, blank out bad sectors, create multiple entries, small files, all without any knowledge of programming! Just \$29.95, plus \$2.00 postage and handling, you become the chief inspector. Sold with the usual Overbeek 30 day money-back guarantee.

MAKE ME A CHIEF INSPECTOR! Enclosed find check for \$31.95 or charge my Mastercard # _____ Expires: _____

Visa# _____ Expires: _____

Check format desired:

<input type="checkbox"/> 8" SSSD	<input type="checkbox"/> Osborne Single Density	<input type="checkbox"/> KayPro II
<input type="checkbox"/> Superbrain	<input type="checkbox"/> Osborne Double Density	<input type="checkbox"/> NEC 5"
<input type="checkbox"/> Northstar Advantage	<input type="checkbox"/> Morrow Micro Decision	<input type="checkbox"/> Televideo802
<input type="checkbox"/> Northstar Horizon	<input type="checkbox"/> Xerox 820 Single Density	<input type="checkbox"/> ALTOS 5
<input type="checkbox"/> Apple/Softcard	<input type="checkbox"/> Xerox 820 Double Density	<input type="checkbox"/> Epson

I'm not ready to order now, but send me information about all the affordable programs from Overbeek Enterprises.

Name _____

Address _____

City _____ State _____ Zip _____

OVERBEEK ENTERPRISES, P.O. Box 726D, Elgin, IL 60120
312-697-8420

CP/M is a trademark of Digital Research.

Disk Inspector — another affordable program from Overbeek Enterprises.

Circle no. 44 on reader service card.

Small-C Library (Listing Continued)

Listing Two

```
6  char *j;
7  int c;
8  j = s + strlen(s) - 1;
9  while(s < j) {
10    c = *s;
11    *s++ = *j;
12    *j-- = c;
13  }
14 }
15
```

File: REWIND.C

```
1 #define NOCCARGC /* no argument count passing */
2 /*
3 ** Rewind file to beginning.
4 */
5 rewind(fd) int fd; {
6   return(cseek(fd, 0, 0));
7 }
```

File: SIGN.C

```
1 /*
2 ** sign -- return -1, 0, +1 depending on the sign of nbr
3 */
4 sign(nbr) int nbr; {
5   if(nbr>0) return 1;
6   if(nbr==0) return 0;
7   return -1;
8 }
```

File: STRCAT.C

```
1 /*
2 ** concatenate t to end of s
3 ** s must be large enough
4 */
5 strcat(s, t) char *s, *t; {
6   char *d;
7   d = s;
8   --s;
9   while (*++s) ;
10  while (*s++ = *t++) ;
11  return(d);
12 }
```

File: STRCHR.C

```
1 /*
2 ** return pointer to 1st occurrence of c in str, else 0
3 */
4 strchr(str, c) char *str, c; {
5   while(*str) {
6     if(*str == c) return (str);
7   }
```

```
7   ++str;
8   }
9   return (0);
10 }
```

File: STRCMP.C

```
1 /*
2 ** return <0, 0, >0 according to
3 ** s<t, s=t, s>t
4 */
5 strcmp(s, t) char *s, *t; {
6   while(*s == *t) {
7     if(*s == 0) return (0);
8     ++s; ++t;
9   }
10  return (*s - *t);
11 }
12
```

File: STRLEN.C

```
1 /*
2 ** return length of s
3 */
4 strlen(s) char *s; {
5   char *t;
6   t = s - 1;
7   while (*++t) ;
8   return (t - s);
9 }
```

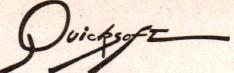
File: STRCPY.C

```
1 /*
2 ** copy t to s
3 */
4 strcpy(s, t) char *s, *t; {
5   char *d;
6   d = s;
7   while (*s++ = *t++) ;
8   return(d);
9 }
```

File: STRNCAT.C

```
1 /*
2 ** concatenate n bytes max from t to end of s
3 ** s must be large enough
4 */
5 strncat(s, t, n) char *s, *t; int n; {
6   char *d;
7   d = s;
```

(Continued on page 68)

PC-Write™ by 

The Capable, Comfortable Word Processor for the IBM PC, PCjr, and Compatible Computers.
Shareware: help us distribute PC-Write by copying and sharing the unmodified diskette!

The Word processor:

Includes wordwrap, search/replace, justify, help screens, block move/copy/delete, headers/footers, bold/underline, many other features for writing and printing of text.

Dr. Dobb's Readers:

PC-Write is a great **program editor!** Fast and easy to use. Features: standard MS-DOS files, jump to numbered line, split screen editing of two files, horizontal scrolling, set bookmark, redefine and record keys, edit in color.

Source Diskette:

IBM/MS-Pascal and assembly. Does DOS 1.2. disk space, EXEC of COMMAND, PATH = file prefix search. Link without the Pascal library and use files, keyboard, and screen. By the MS-Pascal designer. (sources are not shareware).

Programs and manual on diskette:	\$10
Registration (with source diskette, printed manual, support, more):	\$75
VISA/Mastercard: call 206/282-0452 219 First N #224, Seattle WA 98109	

Circle no. 57 on reader service card.

HOW FAST WOULD THIS PROGRAM RUN IF IT WERE COMPILED USING YOUR PASCAL COMPILER ?

PROGRAM SIEVE;
{ THE ERATOSTHENES' SIEVE BENCHMARK }

```
CONST SIZE = 8190;
TYPE BYTE = 0..255;
VAR I, PRIME, K, COUNT, ITER : INTEGER;
FLAGS : ARRAY [ 0..SIZE ] OF BOOLEAN;

BEGIN
  WRITELN( 'START' );
  FOR ITEM := 1 TO 10 DO BEGIN
    COUNT := 0;
    FOR I := 0 TO SIZE DO FLAGS[ I ] := TRUE;
    FOR I := 0 TO SIZE DO
      IF FLAGS[ I ]THEN BEGIN
        PRIME := I + I + 3;
        K := I + PRIME;
        WHILE K <= SIZE DO BEGIN
          FLAGS[ K ] := FALSE;
          K := K + PRIME
        END;
        COUNT := COUNT + 1
      END;
    END;
    WRITELN( COUNT, 'PRIMES' );
  END.
```



LEO ELECTRONICS, INC.

2730 Monterey Street, Suite 111
Torrance, California 90503

(213) 212-6133 • (800) 421-9565
Telex: 291 986 LEO UR

PRICE !

QUALITY !

SERVICE !

RAMS

4116 (150ns)	1.35	16K UPGRADE
4116 (200ns)	1.25	
4164 (150ns)	4.95	
4164 (200ns)	4.75	64K UPGRADE
6116P-3	4.40	

EPROMS

2708	3.00	2532	4.50
2716	3.20	2732	3.95
TMS2716	4.75	2764	7.00

TERMS: Check, Visa, Mastercard. Call for C.O.D.
U.S. Funds only. California residents add 6 1/2% sales tax.

SHIPPING: Add \$2.00 for Ground and \$5.00 for Air.

ALL MAJOR MANUFACTURERS
ALL PARTS 100% GUARANTEED

Pricing subject to change without notice.

Circle no. 34 on reader service card.

Chances are, not as fast as it would if it were compiled using SBB Pascal.

As the following benchmarks show, SBB Pascal outperforms all other Pascal compilers for the PC in terms of speed, code size and .EXE file size:

	Execution Time (secs)	Code Size	EXE File Size
SBB Pascal	10.90	181	4736
MS-Pascal	11.70	229	27136
Pascal/MT+ 86	14.70	294	10752
Turbo Pascal	15.38	288	9029

Development Package
\$350.00

Personal Use Compiler Package
also available
\$95.00

607/272-2807

Software Building Blocks, Inc.
Post Office Box 119
Ithaca, New York 14851-0119

Call for free brochure with full benchmarks.



SBB Pascal is a trademark of Software Building Blocks, Inc. MS-Pascal is a trademark of Microsoft Corporation. Pascal/MT+ 86 is a trademark of Digital Research, Inc. Turbo Pascal is a trademark of Borland International.

Circle no. 68 on reader service card.

Small-C Library (Listing Continued)

Listing Two

```
8  --s;
9  while(*++s) ;
10 while(n--) {
11   if(*s++ == *t++) continue;
12   return(d);
13 }
14 *s = 0;
15 return(d);
16 }
```

File: STRNCMP.C

```
1 /*
2 ** strncmp(s,t,n) - Compares two strings for at most n
3 **                   characters and returns an integer
4 **                   >0, =0, or <0 as s is >t, =t, or <t.
5 */
6 strncmp(s, t, n) char *s, *t; int n; {
7   while(n && *s==*t) {
8     if (*s == 0) return (0);
9     ++s; ++t; --n;
10  }
11 if(n) return (*s - *t);
12 return (0);
13 }
```

File: STRNCPY.C

```
1 /*
2 ** copy n characters from sour to dest (null padding)
3 */
4 strncpy(dest, sour, n) char *dest, *sour; int n; {
5   char *d;
6   d = dest;
7   while(n-- > 0) {
8     if(*d++ == *sour++) continue;
9     while(n-- > 0) *d++ = 0;
10  }
11 *d = 0;
12 return (dest);
13 }
```

File: STRRCHR.C

```
1 /*
2 ** strrchr(s,c) - Search s for rightmost occurrence of c.
3 ** s      = Pointer to string to be searched.
4 ** c      = Character to search for.
5 ** Returns pointer to rightmost c or NULL.
6 */
7 strrchr(s, c) char *s, c; {
8   char *ptr;
9   ptr = 0;
10  while(*s) {
```

```
11   if(*s==c) ptr = s;
12   ++s;
13 }
14 return (ptr);
15 }
```

File: TOASCII.C

```
1 /*
2 ** return ASCII equivalent of c
3 */
4 toascii(c) int c; {
5   return (c);
6 }
```

File: TOLOWER.C

```
1 /*
2 ** return lower-case of c if upper-case, else c
3 */
4 tolower(c) int c; {
5   if(c<='Z' && c>='A') return (c+32);
6   return (c);
7 }
```

File: TOUPPER.C

```
1 /*
2 ** return upper-case of c if it is lower-case, else c
3 */
4 toupper(c) int c; {
5   if(c<='z' && c>='a') return (c-32);
6   return (c);
7 }
```

File: UNGETC.C

```
1 #define NOCCARGC /* no argument count passing */
2 #include stdio.h
3 extern _nextc[];
4 /*
5 ** Put c back into file fd.
6 ** Entry: c = character to put back
7 **         fd = file descriptor
8 ** Returns c if successful, else EOF.
9 */
10 ungetc(c, fd) int c, fd; {
```

```

11 if(!_mode(fd) || _nextc(fd)==EOF || c==EOF) return (EOF);
12 return (_nextc(fd) = c);
13 }

```

File: UNLINK.C

```

1 #define NOCCARGC /* no arg count passing */
2 #include stdio.h
3 #include clib.def
4 /*
5 ** Unlink (delete) the named file.
6 ** Entry: fn = Null-terminated CP/M file name.
7 **           May be prefixed by letter of drive.
8 ** Returns NULL on success, else ERR.
9 */
10 unlink(fn) char *fn; {
11   char fcb[FCBSIZE];
12   pad(fcb, NULL, FCBSIZE);
13   if(_newfcb(fn, fcb) && _bdos(DELFIL, fcb) != 255)
14     return (NULL);
15   return (ERR);
16 }
17 #asm
18 delete equ  unlink
19   entry delete
20 #endasm

```

File: UTOI.C

```

1 #include stdio.h
2 /*
3 ** utoi -- convert unsigned decimal string to integer nbr

```

```

4 **           returns field size, else ERR on error
5 */
6 utoi(decstr, nbr) char *decstr; int *nbr; {
7   int d,t; d=0;
8   *nbr=0;
9   while(((*decstr)>='0')&(*decstr<='9')) {
10    t=*nbr;t=(10*t) + (*decstr++ - '0');
11    if ((t>0)&(*nbr<0)) return ERR;
12    d++; *nbr=t;
13   }
14   return d;
15 }

```

File: XTOI.C

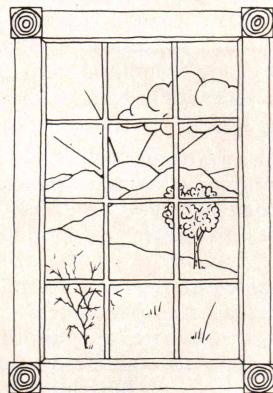
```

1 #include stdio.h
2 /*
3 ** xtoi -- convert hex string to integer nbr
4 **           returns field size, else ERR on error
5 */
6 xtoi(hexstr, nbr) char *hexstr; int *nbr; {
7   int d,t; d=0;
8   *nbr=0;
9   while(1)
10   {
11    if(((*hexstr)>='0')&(*hexstr<='9')) t=48;
12    else if(((*hexstr)>='A')&(*hexstr<='F')) t=55;
13    else if(((*hexstr)>='a')&(*hexstr<='f')) t=87;
14    else break;
15    if(d<4) ++d; else return ERR;
16    *nbr=*nbr<<4;
17    *nbr=*nbr+(*hexstr++)-t;
18   }
19   return d;
20 }

```

End Listings

NOW FOR THE
IBM PC
+ COMPATIBLES



WINDOWS FOR C™

SCREEN MANAGEMENT TOOL FOR C PROGRAMMERS

COMPLETE WINDOW DISPLAY SYSTEM

- Unlimited windows and text files
- Horizontal and vertical scrolling
- Variable text margins
- ASCII file handling

SIMPLIFY • IMPROVE

- Menus
- Help files
- Data screens
- Editors

ALL DISPLAYS

ADVANCED FEATURES

- Instant screen changes
- No snow, no flicker
- Word wrap, auto scroll
- Overlay and restore windows
- Change attribute of selected text

C SOURCE MODULES FOR

menus, cursor control, ASCII file
display, text-mode bar graphs;
+ complete building block
subroutines

Available For: Lattice C, C86, Microsoft C, DeSmet C (MS or PC DOS)

Introductory Special \$119.95
(\$150 after May 31)
Demo disk & manual \$30
(Applies toward purchase)

A PROFESSIONAL SOFTWARE TOOL FROM
CREATIVE SOLUTIONS
21 Elm Ave., Box D5, Richford, VT 05476

For Order or Information: 802-848-7738
Master Card & Visa Accepted
Shipping \$2.50
VT residents add 4% tax

Circle no. 15 on reader service card.

Comments on "Sixth Generation Computers"

The dialogue between Michael Doherty and Richard Grigonis stretches back over the past year and a half. Having begun things in December 1982 with a brief discussion of fifth generation computers, Grigonis held forth last month on what to expect in the sixth generation. Here are Doherty's thoughts in reply. — Ed.

Most readers may recall my previous confrontations with Richard Grigonis in the pages of *Dr. Dobb's Journal* and how ridiculous he made me look with his article "And Still More Fifth Generation Computers" in August 1983. Naturally, it was with a feeling of great foreboding that I recently opened an envelope bearing the corporate logo of the Children's Television Workshop, as it could mean only one thing: Grigonis was sending me an advance copy of his latest article ("Sixth Generation Computers") and was daring me to do something about it!

Upon reading the article, however, I was relieved to discover that Grigonis's vast intellect had finally confounded itself and that I may yet emerge victorious from a final confrontation. Grigonis doesn't make it any easier on himself when he writes that the favorable response to his last article has encouraged him to write yet another opus, "taking speculation about future computers to the limits that can be conceived of by the human mind." His modest claims remind me of the tale of Erasmus from those strange post-Biblical writings known as the Apocrypha. The story goes that Erasmus, most learned man on Earth, wishes to demonstrate that, like Jesus, he can ascend even unto heaven, and so he begins to float up into the sky. Saint Peter, appalled at this flagrant display, prays to God that something be done, whereupon Erasmus falls to Earth and breaks his arms and legs. Saint Peter then breathes a sigh of relief and sternly notes to the reader that "here was a man who could fly like a bird, and now he cannot even crawl like a worm."

by Michael Doherty

Grigonis's predictions this time around include reducing the execution time of artificial intelligence (AI) programs by building processors having signals that can travel faster than light, which incidentally allows the computer to answer questions even before the programs asking the questions are run! His other prediction is that we will be able to communicate with such "superluminal" sixth generation computers through a technique whereby the computer reads the user's mind! These ideas definitely beg closer scrutiny, so let us continue.

First of all, most physicists at one time would have agreed with Grigonis's statement that "there is nothing in relativity theory, quantum mechanics, electrodynamics, or mechanics that says that time must move in one direction," but a little item from quantum mechanics may one day prove otherwise: namely, the controversy over the conservation of charge conjugation, parity reflection, and time reversal, the so-called "CPT theorems" initially stated by Wolfgang Pauli in 1942 and proved by the Swiss physicist Res Jost in 1958. The strong possibility exists that certain subatomic processes can move only forward in time. On the other hand, violations of the conservation laws may be possible, and we therefore might have to throw out most of physics, but this is unlikely. The whole question is still up in the air and is certainly not as open-and-shut as Grigonis states.

Physicist Bryce S. DeWitt, however, writing in a recent issue of *Scientific American*, discusses the idea of "quantum gravity" and how a subatomic event that is quick enough, or of a high enough energy, could reveal the quantum "grainy" structure of space-time, thereby blurring the distinction between the past and future.¹ In more scientific terms, if gravitation (and, hence, space-time) is quantum in nature, then the shape of the light cone defining the regions of space-time that are accessible from a given point in space and moment in time would act as if it were "fuzzy." Therefore, the ordinary accepted phenomenon of two points in space-time communicating with each other by means of signals moving equal to or less than the speed of light can be given only a probabilistic certainty. At tiny Planckian dimensions (1.61×10^{-33} centimeter), at Planckian time units (5.36×10^{-44} second), and at high enough energies, the probability of a signal traveling faster than the speed of light between two points

could be very high indeed or at least high enough to make one of Grigonis's superluminal processors possible.

But to test Grigonis's theory, the energies required to explore this sub-subatomic realm would be immense. As DeWitt writes in his article: "To probe these scales of distance and time experimentally, using instruments built with present technology, one would need a particle accelerator the size of the galaxy!" I somehow doubt that even IBM would want to build one of Grigonis's proposed sixth generation computers if it entailed having to construct superluminal processor switches the size of the Milky Way!

Grigonis tries to get around this with all sorts of ingenious arguments that hover at the most distant horizon of theoretical physics. I would like to comment on them but I don't feel that I'm qualified. Indeed, I can think of only a handful of physicists in the world who are capable of seriously evaluating some of the things he suggests, which makes me wonder how he thought of them himself!

As for the "mind-reading" aspects of his proposed computer, I also find these difficult to disprove, but only because the physics is once again beyond my level of training. In fact, I have a sneaking suspicion that Grigonis may even be right! Richard Feynman and Steven Weinberg, where are you when we need you?

Moreover, Grigonis has reduced all of AI to a matter of searching through a search space that consists of the logical relationships of every known fact or assertion. Instead of resorting to forms of constrained, heuristic search, Grigonis astonishes us with his suggestion to stay with the "brute force" method and simply build computers with processors having signals moving faster than the speed of light.

Grigonis warns us that such superluminal computers would answer our questions long before we ever ask them. He proposes to solve this problem by placing every query in a buffer where a conventional computer and AI program "would analyze the parameters of the question and estimate the appropriate amount of time to wait before sending the question to the superluminal processor." Grigonis goes on to assure us that, "if the statement was accurate, the delay period of submitting the question to the superluminal processor would be just a little longer than the negative time required for an answer . . ." The answer

thus would appear immediately after you enter the question instead of sometime in the past.

The problem with making a "rough estimate" of the execution time of any program is that, because of such things as Gödel's Proof and Turing's Halting Problem, such an estimate can only be rough, never precise: there is no way to determine whether a Turing machine (digital computer) is ever going to stop running any particular program. As Gödel discovered, there are "undecidable propositions" in logic; certain problems take an infinite amount of time to solve because they are unsolvable, and no logical way exists to determine if the computer is ever going to stop. All of this means that, if one of Grigonis's sixth generation computers were studying a really difficult, "interesting" problem, then the estimate of how long to hold the query in the buffer would certainly be off by at least an order of magnitude, and the answer could appear unpredictably in the past, present, or future!

Grigonis ignores another plan that does not require a superluminal processor, but here too the argument breaks down. I am referring to the constant trade-off of processing time versus storage size. Instead of using simple assertions and a resolution problem solver, one could make the knowledge representation of a particular problem more powerful by storing large, complex frames or semantic nets on disk.

For example, Grigonis is correct when he says that, given an initial chess position, up to 10^{150} move sequences are possible, requiring eons of processing time to search through. However, let's imagine that we've already calculated every possible move in every possible chess game *once* and have stored them all on a gigantic disk drive in the form of a series of records or vectors. With such an extreme form of knowledge representation, the processing time becomes almost zero as the problem of making the next move is now reduced to looking up entries in a table; the program simply looks up the current configuration of chess pieces on its internalized "chart" and immediately sees the entry that tells it the best move to make next. Deduction as such would be unnecessary, since this or any other problem situation could in theory be made complete in terms of all relevant objects, properties, and relations.

Of course, we have traded one impossibility for another: instead of an infinite amount of processing time, we now must provide an infinite number of disk drives! Besides, one rarely has a complete description of any problem situation, and a representation formalism based on logic does make it possible to express generalizations economically in spite of this. Still, just because we can deduce the behavior (the

various possible states) of a system from its description as a set of subsystems or even simple assertions does not imply that the system can be *explained* from any kind of modular, logical, representation/resolution, theorem-proving process. AI researchers had hoped to develop logical, problem-independent solution methods, but we now know from meta-mathematical grounds that no general, efficient, problem-independent solution technique exists.^{2,3}

And what if the fundamental concepts themselves cannot be broken down into their component features with any utility? E. Rosch,^{4,5} K. Nelson,⁶ and

D. S. Palermo,⁷ just to name a few, have complained that the feature theory of semantic development in human language is wrong, that things are more complicated than just applying good old reductionism to the problem of meaning.

In feature theory, features are abstracted from a whole concept so that the concept's meaning can be known. But one must already know that whole concept in order to know what features to abstract from it. Also, as Nelson says, like Humpty Dumpty there is no way of putting the abstracted features together to form the original whole. As Palermo points out, "a list of features is not a concept."

PRICE BREAKTHROUGH

The wait-loss experts have done it again!

512Kbyte SemiDisk™ with SemiSpool™

\$1095

Time was, you thought you couldn't afford a SemiDisk. Now, you can't afford to be without one.

	256K	512K	1Mbyte
SemiDisk I, S-100	\$895	\$1095	\$1795
IBM PC		\$1095	\$1795
TRS-80 Mdl. II, CP/M		\$1095	\$1795
SemiDisk II, S-100		\$1395	\$2095
Battery Backup Unit	\$150		
Version 5 Software Update		\$30	

Time was, you had to wait for your disk drives. The SemiDisk changed all that, giving you large, extremely fast disk emulators specifically designed for your computer. Much faster than floppies or hard disks, SemiDisk squeezes the last drop of performance out of your computer.

Time was, you had to wait while your data was printing. That's changed, too. Now, the SemiSpool print buffer in

our Version 5 software, for CP/M 2.2, frees your computer for other tasks while data is printing. With a capacity up to the size of the SemiDisk itself, you could implement an 8 Mbyte spooler!

Time was, disk emulators were afraid of the dark. When your computer was turned off, or a power outage occurred, your valuable data was lost. But SemiDisk changed all that. Now, the Battery Backup Unit takes the worry out of blackouts.

But one thing hasn't changed. That's our commitment to supply the fastest, highest density, easiest to use, most compatible, and most cost-effective disk emulators in the world.

SemiDisk.

It's the disk the others are trying to copy.

SEMDISK SYSTEMS, INC.

P.O. Box GG Beaverton, OR 97075 (503) 642-3100

Call 503-646-5510 for CBBS/NW, a SemiDisk-equipped computer bulletin board. 300/1200 baud
SemiDisk, SemiSpool Trademarks of SemiDisk Systems CP/M Trademark Digital Research



NO WAITING

Circle no. 63 on reader service card.

Names or words referring to categories are acquired as a whole and not feature by feature. They are highly structured internally but paradoxically do not have sharp, distinct boundaries; therefore, things are classified not by their features but by their "distance" from an ideal prototype or "core exemplar," which sounds suspiciously like Plato's old philosophy of ideals and his doctrines of *form* and *anamnesis*. Rosch shows that categories are universal across languages because they are intrinsic biological structures given genetically to the brain from its evolutionary heritage.

All of this would explain why, like the conceptual dependency theory of semantic primitives in AI programs, the feature theory of human semantic acquisition ignores the existence of metaphors, even though it is almost impossible in practice to differentiate between metaphors and nonmetaphors. We can speak nonmetaphorically of "the long arm of the man," but we can also easily comprehend "the long arm of the crane," or "the long arm of the law." All words have some metaphoric power, the ability to overlay two different objects or events. As Marvin Minsky has written: "What something means to me depends on everything else I

know... every meaning is built on other meanings, with no special place to start."⁸

Similarly, it also seems impossible to assign items to a finite number of categories and then state rules in terms of membership in those categories. John Ross, for instance, observed that certain verb structures could generally be used in place of a noun phrase with varying degrees of acceptability.⁹ For example, the gerund "his going" can be used in the same positions as a normal noun phrase:

I regretted *his going*.

His going surprised me.

After *his going* we hired a new worker.

His going's main effect was to end the feud.

Ross thought that properties like "nouniness" could be placed along a scale he called a *squish*. His attempts to devise a rigorous formalism for squishes have not met with success, and AI researchers in the natural language processing field tend to sweep them under the rug.

So it seems that a computer program can have only a syntax – no real semantics. Of course these squishes, like the other feature/category problems, could be explained if the human brain were viewed not as a discrete machine like a Turing machine or a digital computer but as a

series of parallel analog devices.

This brings us to some fundamental problems that Grigoris ignores. Since a universal Turing machine can simulate any calculating device or physical expression of any algorithm, and assuming that the mind arises out of the physical workings of the brain, it follows that the physical embodiment of a Turing machine (the digital computer) can simulate human cognition.

But what if the human brain is not a Turing machine? What if it is some kind of nondiscrete analog device? This would explain how it can comprehend continuous spectra of physical state changes without analyzing each discrete state. Or worse yet, what if the brain is a series of parallel analog processors?

John Searle exposed an even more fundamental misconception among AI researchers when he demonstrated that cognition is not solely a matter of formal symbol manipulation.¹⁰ Most AI researchers think that "the mind is to the brain as the program is to the hardware," which implies that a Turing machine in the form of a program running on a digital computer can achieve cognition. This ignores the fact that, while even primitive AI programs can do things that people can do, they utilize only simple computations that have nothing to do with the kind of actual cognitive states experienced by humans.

After all, the strange fact about AI has always been that it's easier to simulate an expert than to simulate the general common sense of a five-year-old. Daniel Bobrow's STUDENT program could solve high school algebra word problems, and James Slagle's SAINT and Joel Moses's SIN programs could solve rather complex, abstract calculus problems. Amazingly, these programs relied on only about a hundred rule-based "facts" to achieve their mathematical miracles, whereas Terry Winograd's SHRDLU program, designed to manipulate a simple world of children's building blocks, was over 80,000 words long and required many more rules than most of the "expert" systems!

As Searle says, "simulation is not duplication." A person with a photographic memory, after all, could memorize a program and "think it through," immediately realizing that no intentional states are involved, just a set of instructions having nothing to do with "understanding." As Searle writes: "No one supposes that computer simulations of a five-alarm fire will burn the neighborhood down or that a computer simulation of a rainstorm will leave us all drenched. Why on earth would anyone suppose that a computer simulation of understanding actually understood anything?" AI researchers are still under the spell of the Turing Imitation Game, with its strong smell of behaviorism and operationalism.

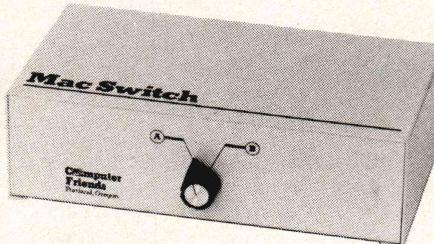
Re-ink any fabric ribbon for less than 5¢. Extremely simple operation. We have a MAC INKER for any printer. Lubricant ink safe for dot matrix printheads. Multi-colored inks, unlinked cartridges available. Ask for brochure. Thousands of satisfied customers.

\$54.95 +



Mac Switch lets you share your computer with any two peripherals (serial or parallel). Ideal for word processors—never type an address twice. Ask us for brochure with tips on how to share two peripherals with MAC SWITCH. Total satisfaction or full refund.

\$99.00



Computer Friends

6415 SW Canyon Court
Suite #10
Portland, Oregon 97225
(503) 297-2321

Order Toll Free 1-800-547-3303

Mac Inker & MacSwitch

Circle no. 12 on reader service card.

Ironically, the idea that cognition can be expressed in a set of logical transformations — a computer program — set apart from the hardware, be it computer or brain, implies a strange mind/body dualism that goes against the avowed materialistic stance of AI researchers. Suppose the biochemical reactions of the brain itself are responsible for cognition? As Hubert L. Dreyfus has pointed out, knowledge representation with formal symbols and transformations may be unnecessary in human cognition: "... these are usually nonformal representations, more like images, by means of which I explore what I am, not what I know. We thus appeal to concrete representations (images or memories) based on our own experience without having to make explicit the strict rules and their spelled out *ceteris paribus* conditions required by abstract symbolic representations."¹¹

So although humans are sometimes annoyingly imperfect thinkers, at least they do think, unlike AI programs and the computers that run them. The fundamental workings of the universe since the Big Bang seem to legislate against building one of Grigonis's "ultraintelligent machines."

In closing, let me thank *Dr. Dobb's Journal* for allowing me to once again

continue the "Grigonis-Doherty debate" within its pages. I can think of no other publication in this country with a readership capable of appreciating such esoterica. And, despite appearances to the contrary, I would also like to thank the Grand Master of Computerdom himself, Richard Grigonis, for writing the most brilliant and daring series of articles of the past 30 years, or at least since the legendary days of Alan Turing and Johnny von Neumann.

References

- 1 DeWitt, Bryce S. Quantum gravity. *Scientific American*, Dec. 1983, Vol. 249, pp. 112-129.
- 2 Davis, M. *Computability and Unsolvability*. New York: McGraw-Hill, 1958.
- 3 Rabin, M. Theoretical impedance to artificial intelligence. *Proc. Conf. Inter. Fed. Info. Processing Societies*, 1975, Stockholm.
- 4 Rosch, E. On the internal structure of perceptual and semantic categories. In T. E. Moore (ed.), *Cognitive Development and the Acquisition of Language*, New York: Academic Press, 1973, pp. 111-144.
- 5 Rosch, E. Cognitive representation of semantic categories. *Journal of Experi-*

mental Psychology, 1975, Vol. 104, pp. 192-233.

6 Nelson, K. Concept, word, and sentence: Interrelations in acquisition and development. *Psychological Review*, 1974, Vol. 81, pp. 267-285.

7 Palermo, D. S. Semantics and language acquisition: some theoretical considerations. Paper presented at the Psychology of Language Conference, University of Stirling, June 1976.

8 Minsky, Marvin. Why people think computers can't. *Technology Review*, Nov./Dec. 1983, Vol. 86, pp. 65-81.

9 Ross, John R. *Constraints on Variables in Syntax*. Ph.D. dissertation, MIT, 1967.

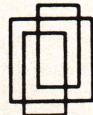
10 Searle, John R. Minds, brains, and programs. In John Haugeland (ed.), *Mind Design*, Cambridge: The MIT Press, 1981, pp. 282-306.

11 Dreyfus, Hubert L. From micro-worlds to knowledge representation: AI at an impasse. In John Haugeland (ed.), *Mind Design*, Cambridge: The MIT Press, 1981, pp. 161-204.

BBJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 197.



LATTICE® C Compilers

"My personal preferences are Lattice C in the top category for its quick compile and execution times, small incremental code, best documentation and consistent reliability..."

BYTE AUG. 1983

R. Phraner

"...programs are compiled faster by the Lattice C compiler, and it produces programs that run faster than any other C compiler available for PC-DOS."

PC MAGAZINE JULY 1983

H. Hinsch

"...Microsoft chose Lattice C both because of the quality of code generated and because Lattice C was designed to work with Microsoft's LINK program."

PC MAGAZINE OCT. 1983

D. Clapp

"Lattice is both the most comprehensive and the best documented of the compilers. In general it performed best in the benchmark tests."

PERSONAL COMPUTER AGE NOV 1983

F. Wilson

"This C compiler produces good tight-running programs and provides a sound practical alternative to Pascal."

SOFTALK AUG 1983

P. Norton

"...the Lattice compiler is a sophisticated, high-performance package that appears to be well-suited for development of major application programs."

BYTE AUG 1983

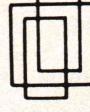
Houston, Brodrick, Kent

To order, or for further information
on the LATTICE family of compilers, call or write:



LATTICE, INC.
P.O. Box 3072
Glen Ellyn, IL 60138

(312) 858-7950 TWX 910-291-2190



Circle no. 33 on reader service card.

CDE SOFTWARE

DISK FORMAT CONVERSION SERVICES

CDE now offers disk conversions from one format to another. We can even convert text files from MS-DOS to CP/M and back.

The cost? Only \$7.00 per disk, and we supply the destination disk! (\$5.00 if you supply the disk.) Additional copies of the same disk are only \$5.00 each (\$3.50 on your disk). If a second disk is required to hold down-loaded data, it is \$2.00 additional. VISA and MasterCard accepted.

Formats available include 8" SSSD and the following 5" formats:

Kaypro	ACCESS
Osborne	LOBO MAX-80
Xerox 820	TI Professional
TRS-80 mod-I (Omkron CP/M)	HP-125
TRS-80 mod-III (MM CP/M)	Televideo TS-802
IBM-PC for CP/M	Otrona
Morrow	IMS-5000
NEC PC-8001A	EPSON QX-10
Zenith Z-90	Sanyo
Heath with Magnolia	NEC PC-8801
Superbrain	Zenith Z-100
Cromemco	Datavue
DEC VT-180	MAGIC Computer

Write for our complete catalogue.

2463 McCready Avenue • Los Angeles, CA 90039

Circle no. 9 on reader service card.

How many times have you said to yourself, "What this industry (still) needs is a good, low-cost Pascal compiler"? We've said the same thing ourselves, and we were fascinated to learn about Turbo Pascal. Not only low in price, it was advertised to be fast, take very little space, and include a resident editor. In addition, from everything we can determine, Borland ships promptly and provides good support. They have even disposed of their \$100 commercial licensing fee.

Sound too good to be true? Well, the following review suggests that Turbo Pascal is what it claims. By the time you read this (lags in publishing being what they are) you will probably have seen ads in DDJ for a version 2.0 that provides even more features, including automatic overlays, windows (at least for IBM PC and PC Jr.), and a dispose function. We will provide an update on that newer version in the next few months. In the meantime, Borland deserves watching. With similar packages for Modula-2 and C coming soon, their potential impact should not be underestimated. — Ed.

Turbo Pascal, V.1.01

Company: Borland International, 4807
Scotts Valley Drive, Scotts Valley,
CA 95066

Computer: CP/M-80, MS-DOS, or
CP/M-86

Price: \$49.95

Circle Reader Service No. 119

Reviewed by David D. Clark

The advertisements say that "This is the Pascal compiler everybody's been waiting for." They may be right. Turbo Pascal, from Borland International, is an excellent product at an extraordinary price. Some people may be leery of a low-priced Pascal compiler after having had a bad experience with JRT Pascal. I myself had similar feelings when I saw the advertisements for Turbo Pascal. Since using the product, though, I have been pleasantly surprised. *Amazed* might be a better word.

When you open the package, you find a disk and a reference manual. The disk contains several programs and text files. TURBO.COM (or TURBO.CMD), TURBOMSG.OVER, and TURBO.OVR make up the compiler/editor, error messages, and a temporary program loader. Also present are an installation program and associated data file, a program-

listing utility, an errata file, and the source files for a simple spreadsheet program called MicroCalc.

Turbo Pascal has many interesting and innovative features. It is very small, only about 28K. A full-screen editor is co-resident with the compiler. The compiler is exceptionally fast. Versions are available for both 8-bit and 16-bit systems. The user can develop highly standard, portable programs. The programs are reasonably fast. A floating-point format with eleven decimal digits allows business applications to handle dollars and cents calculations to \$999,999,999.99. Access is provided to the underlying hardware and operating system facilities. In-line machine language can be generated, and the user can control compiler operation with a variety of directives.

The Editor

The built-in, full-screen editor is one of the major conveniences this package provides. It is co-resident with the compiler and, as such, allows a speed of interaction during program development that is unprecedented with a compiled language. It is possible to enter the editor part of the package, create the source text for a program, exit from the editor, compile the text in memory, and immediately return to the editor in the event of a compilation error, all without any disk access required. It's fast.

The first thing you will want to do after scanning the reference manual is install the editor for your computer. This is done by running the installation program TINST. There are two parts to the process: terminal installation and command installation. The first and most important is installing the correct terminal driver for your CRT. TINST uses a data file containing information about a number of common terminals. If yours is one of those already known to the system, you will just have to select it from a menu and the program will finish the installation. If your terminal is not one of those listed, or if you wish to change the default parameters for a listed terminal, you will be led through a series of questions that will set up the system to run with your equipment. I tried the installation procedure with three different terminals. The system had menu entries for two of them, a Zenith and a TeleVideo, and installed them without a hitch. I also installed the system on a North Star Advantage, which is not listed

in the menu. The installation process was clear and quick — you just need to know the control codes for your particular terminal.

Turbo Pascal has several procedures that are extensions to standard Pascal and that allow control of the video screen of your terminal. You must complete the installation process correctly in order to guarantee that these procedures work as expected.

The default commands for the editor are a subset of those used by WordStar. Cursor movement, insertion, deletion, find, replace, and the block commands are almost identical with those of WordStar. Some additional commands are also present. The editor's differences from WordStar, such as its incorporation of automatic indentation and a single command to mark a word as a block, make the editor easier to use in a program-development environment than WordStar. If you know WordStar, you will be able to use the Turbo editor.

The installation program also allows you to change the characters used to invoke each command. If you don't like the default choices, it's easy to alter them. Just remember to write down your changes somewhere.

Running Turbo Pascal

The heart of the system consists of the program TURBO. While running, TURBO is a simple program-development environment. When you first start TURBO, it informs you of the terminal it has been installed for and asks if you wish to include error messages. If you respond with an affirmative, a text file containing the compiler error messages will be loaded. These contain the text that will be displayed in the event the compiler detects an error while translating a program. If you do not load the error messages, you will have about 1.5K additional free memory, but the compiler will just print out a number when it detects an error. After the error messages are loaded (or not), a menu is displayed on the screen. The available options are shown in Table 1 (page 75).

Speed

A rather large demonstration program is included as a group of source files on the distribution disk. It is a simple spreadsheet program called MicroCalc. The first thing I did after backing up the master disk (even before installing

the editor) was compile this program. That was when I got my first surprise. The 1261-line program was compiled to a 22K object file in almost exactly one minute. This is just about the fastest compilation speed I have ever seen on an 8-bit microcomputer.

When compilations are done in memory (selected with the Options menu), the process is even faster because there are no disk-write accesses to slow things down. It is a truly impressive thing to see. I ran a couple of benchmarks to see how fast the generated code is. The first was the (in)famous *Byte* sieve prime-number generator. The program compiled almost instantly in memory, too fast for me to get an accurate measurement by hand. It generated 301 bytes of code (not including the data space occupied by the large Boolean array). The program executed in 24 seconds. Although this is not the fastest execution speed, it isn't too shabby either. It is much faster than UCSD or JRT Pascal and several seconds faster than the C version of the program when compiled with Aztec C II or Eco-C on my system.

Another benchmark I tried was the floating-point program from *Dr. Dobb's* No. 83 and No. 89. Because Turbo Pascal does not include a standard function for calculating the arctangent, the program declared a function to calculate one. This program took 337 seconds to execute (again compilation was almost instantaneous) and had an error of 4.6E-3 from an exact answer of 2500. This is not too bad for software floating point on an 8-bit machine, especially when you consider that the floating-point format in Turbo Pascal uses a 40-bit mantissa (about 11 significant decimal digits), compared to 24 bits (about 7 significant decimal digits) in normal single-precision.

The Reference Manual

The reference manual is a paper-bound book of about 260 pages. The main part of the manual describes the features common to both the 8-bit and 16-bit versions of Turbo Pascal. The first two appendices, each about 30 pages, describe those parts of the system, specific to either the 8- or 16-bit version. The appendix describing the 16-bit version is further divided into sections covering the MS-DOS version and the CP/M-86 version. There are also a number of additional appendices that list the standard procedures and functions, operators, compiler directives (described below), differences from standard Pascal (also described below), compiler error messages, runtime error messages, I/O error messages, translation of error messages to foreign languages, detailed installation procedures, Turbo Pascal syn-

tax, the ASCII character set, and a subject index.

Although the manual states that it is not intended to teach Pascal, it does cover the language features thoroughly with lots of examples. Unfortunately, there are some rough spots. There is a tendency to use *e.g.* with annoying frequency and often at an inappropriate place in a sentence. The same thing occasionally happens with *i.e.* A word processor appears to have been allowed free rein with decisions on where to hyphenate words. For example, "sc-reen" and "th-rough" are two of the more blatant blunders.

These errors aren't really too important. Every book of this size has some typographical or usage errors. Even reviews such as this one are not immune. *Real* confusion can occur when a program example is in error or does not agree with the surrounding text. For

example, in the discussion of operations on text files, an example program declares a variable of type Text named "F." In the body of the program, however, all text-file operations are performed on an undeclared variable "FilVar."

If there is a weak point in the system, it is the reference manual. It isn't too bad, but it isn't anything great either. Mostly it's just sloppy. It needs one more thorough edit.

Deviations from Standard Pascal

The degree to which a particular implementation of a programming language adheres to the language standard affects the portability of programs developed with that implementation. Turbo Pascal uses the *Pascal User Manual and Report* by Kathleen Jensen and Niklaus Wirth, *not* the ISO standard, as the definition of standard Pascal. For most users,

Logged drive	Allows you to change the currently logged drive.
Work file	The work file is the file maintained in memory for alterations with the editor.
Main file	This command is used when the work file contains a file that is included from some other main program. If you later elect to compile a program, the work file will be saved and the main file will be read into memory for compilation.
Edit	Takes you to the editor to create or alter the work file.
Compile	Compile the work file or main file if different from the work file. A successful compilation can result in a "COM" (or "CMD") file, a "CHN" file, or a memory resident program, depending upon the selections made with the Options menu item.
Run	This command will run a memory resident or executable file. If the program has not been compiled, the compiler will be invoked automatically.
Save	Used to save an altered work file back to disk. Any previous version is renamed with the file extension "BAK" before the new version is saved.
Execute	Will run another program from TURBO. When the program terminates, control is transferred back to the TURBO program.
Directory	Allows viewing the contents of a disk directory. Disk specifiers and wild cards are allowed.
Quit	Exits the TURBO program. If the work file has been altered but not saved, you will be asked if you want to store it before leaving the program.
Options	This command varies slightly in the 8-bit and 16-bit versions. It allows you to direct the compiler to construct a program in memory or to disk. If a disk file is selected, it can be one of two types: an executable "COM" (or "CMD") file or a "CHN" file. A "CHN" file can be "chained" to or from another Turbo Pascal program. It does not contain the runtime library because one will be present from the calling program. Because no library is present, "CHN" files are about 8K smaller than executable files. It is also possible to specify the starting address of a program and the last memory location available to the program with the 8-bit version. The 16-bit version allows the user to specify a minimum code-segment size, a minimum data-segment size, and the minimum and maximum allowable dynamic memory.
	The options command also allows you to find where a runtime error occurred in a "COM" or "CHN" file.

Table 1

the difference between the two standards will be small, the largest difference being the inclusion of conformant arrays in the ISO standard. I'm not aware of a microcomputer-based Pascal that implements conformant arrays yet, anyway.

One of the appendices of the Turbo reference manual discusses deviations from the Jensen and Wirth standard. These fall into (basically) six areas and are mostly concessions to efficiency; that is, abandoning facilities that are difficult to implement. The deviations discussed do not include the many extensions available in the Turbo version. The deviations are:

1. The standard procedure **Dispose** is not available. Instead, you can manage heap space (used for dynamic variable allocation) by use of the

standard procedures **Mark** and **Release**. The dynamic-variable allocation procedure **New** is also restricted; the Turbo version does not allow variant record specifications.

2. The standard I/O (input/output) procedures **Get** and **Put** are not implemented. Instead, the **Read**, **ReadIn**, **Write**, and **WriteIn** have been extended to handle all I/O.
3. The **Goto** statement is restricted in that it may transfer control only within the current block.
4. The standard procedure **Page** is not implemented.
5. Variable packing and unpacking are not under user control. Variables are packed whenever possible. Because of this, the reserved word **packed** has no effect and the standard proce-

dures **Pack** and **Unpack** are not implemented.

6. A program may not pass procedures and functions as parameters to other subroutines.

The discussion of the **with** statement in the main body of the reference manual would also lead you to believe it is nonstandard. According to the manual, the user must dereference successive fields within records with commas instead of with periods as in standard Pascal. The number of records that the user can open using a **with** statement is also restricted, but the user can change that number by use of the "W" compiler directive. The version of the compiler I used was capable of handling the standard syntax as well. In fact, on all the tests I ran, using the standard form generated slightly less code than the technique presented in the reference manual.

Finally, the error messages are not the same as in the *Pascal User Manual and Report*. This is not really a standard, but most compilers based on the P4 portable compiler, developed by Professor Wirth's colleagues in Zurich, do use the same set of error messages.

I transferred several programs written for the UCSD version of Pascal to CP/M for use with Turbo Pascal. In most cases, the revisions needed to get the programs running under Turbo Pascal were minor, usually involving things that the UCSD system implements in a nonstandard manner, like opening files. I did have some difficulty in converting programs that made extensive use of the **Get**, **Put**, and **Dispose** standard procedures.

Extensions

Turbo Pascal provides a large number of extensions to standard Pascal. They usually support a simple means of doing something that is clumsy or impossible within the constraints of the standard language — for example, one provides direct access to the underlying hardware. These extensions may be convenient, but programs developed with them will be less portable than those that use only the facilities available in the standard.

The extensions available are numerous, so I will briefly summarize some of them. To paraphrase from the user's manual, they include:

1. Dynamic strings. Strings are almost indispensable for many programs. Most implementations of Pascal on microcomputers provide strings of some sort, so they are almost a standard. Turbo Pascal strings, and standard procedures and functions to manipulate them, closely resemble those available in other implementations such as JRT Pascal and UCSD Pascal.

Table 2.

2. Variable declarations specifying an absolute address to be occupied by the variable.
3. Bit and byte manipulation.
4. Direct access to memory, data ports, and operating-system facilities.
5. Relaxed restrictions on the ordering of **const**, **type**, and **var** declarations.
6. Ability to generate in-line machine code.
7. File inclusion from the main program.
8. Logical operations on integers, such as left and right shifts.
9. Program chaining with common variables.
10. Random-access data files.
11. Structured constants that allow a form of initialized variable declaration.
12. Type-conversion functions.
13. A large number of built-in routines to allow control of video terminals.

Extensions have also been made to the syntax of some statements. For example, the **case** statement allows an optional **else** part and subranges in **case** labels.

Compiler Directives

The user can control the operation of the compiler by use of several compiler directives included in programs as a special form of comment. There is a group of directives common to both the 8-bit and 16-bit versions of the system, as well as some present only on the 8-bit or 16-bit versions. These are shown in Table 2 (page 76).

There is only one directive specific to the 16-bit versions of Turbo Pascal. When the "K" directive is active, as it is initially, a check is made for adequate stack space before each subroutine call. When the directive is turned off, no such checks are performed.

Chaining and External Procedures

There are no true separate compilation facilities available with Turbo Pascal. Because compilation is so fast, however, it is almost as easy to develop files of commonly used routines and include them in the compilation of the main program as it would be with a separate-compilation feature. Except for very large packages of routines, this should be sufficient for most needs.

Also, there are no program-overlays or -segment facilities. It is possible to compile a program of type "CHN" that can be called from another Turbo Pascal program with the **Chain** standard procedure. A "CHN" file does not contain a copy of the Turbo Pascal runtime package and is, therefore, smaller than a "COM" file would be. In addition, it is possible to share variables between the calling program and the program chained

to. You can use the **Execute** procedure to start execution of a "COM" file from a Turbo Pascal program. Both of these procedures place a value of FF (hex) in the command-line buffer as a means of notifying the called program that it has been invoked from another program. Because of this, calls should not be made to programs that expect to make use of information from the command line.

The use of external procedures is a little tricky in Turbo Pascal. It doesn't appear to be possible at all with programs compiled in memory. With a minor deviation, external procedures and functions are declared in a manner similar to the

declaration of a **forward** subroutine; the procedure or function heading is specified, followed by the **external** reserved word. Following the **external** reserved word, there must be an absolute address for the routine.

Turbo Pascal assumes that all **external** subroutines will be written in assembly language. The reference manual clearly explains the parameter-passing and function-return protocols needed to make such assembly-language routines work with Turbo Pascal.

When the program is compiled, the starting address must be set explicitly from the Options menu to make room

SAVE YOUR 8 BIT SYSTEM WITH THE ONLY TRUE 16 BIT CO-PROCESSOR THAT HAS A FUTURE



DO NOT BUY INTO OBSOLESCENCE LET HSC "STEP" YOUR 8 BIT SYSTEM INTO THE 16 BIT REVOLUTION THROUGH EVOLUTION.

- Easily attaches to ANY Z80 based microcomputer system. Successful installations include: Xerox I&II, Osborn I, DEC VT180, Zenith, Heath, Bigboard, Ithaca, Lobo, Magic, Compupro, Cromemco, Teletek, Altos 8000, Lanier EZ1, Zorba, Morrow, Kaypro, Televideo, etc.
- Dynamically upgrades a CPM-80 system to process under CP/M-86, MS-DOS (2.11), and CP/M-68K with no programming effort. (CCP/M-86 (3.1) and UNIX will be available soon).
- All 16 bit operating systems can use the un-used portion of C0-16 memory as RAM DISK.
- TRUE 16 BIT PROCESSOR SELECTION - 8086 (field upgradable to 80186), 80186, and 68000 (all 6mhz with no wait states - 16 bit data path). **Which spells much higher performance than 8088.**
- Available in a self contained attractive Desktop Enclosure (with a power supply), or in PC Card form for inclusion in 8 bit system. **YOU DON'T HAVE TO CRAM IT INTO YOUR BOX / IF YOU DON'T WANT TO.**
- Does not disturb the present 8 bit operating environment.
- Memory expansion from 256K to 768K RAM.
- Optional 8087 Math Co-Processor on the 8086, and up to FOUR (4) National 16081s on the 68000!!!
- MS-DOS and CP/M may co-share common data storage devices (such as hard disk).

- * MS-DOS Compatible
- * IBM PC "Hardware" Compatible
- * CP/M-86 Compatible
- * CCP/M-86 Compatible
- * CP/M-68K Compatible

- Direct MS-DOS and PC-DOS formatted 5 1/4" Diskette read/write capability available on: Osborn I, Morrow, Kaypro, Televideo 803, and Epson QX-10 systems. More coming.
- All 768K can be used as high speed CP/M-80 RAM DISK.
- Optional Real Time Clock, DMA, I/O Bus, and 2 Serial Ports.
- I/O MODULE CONTAINING AN IBM COMPATIBLE BUS (4 slot) & an IBM COMPATIBLE KEYBOARD INTERFACE is an available option. THIS IS THE REAL DIFFERENCE BETWEEN MS-DOS and IBM PC "HARDWARE" COMPATIBILITY.

AFFORDABLE PRICES

C01686 - includes 8086, 256K RAM, Memory Expansion Bus, Z80 Interface, CP/M-86, MS-DOS (2.11), MS-DOS RAM Disk, CPM-80 RAM Disk. \$650.00

C01686X - includes all of C01686 PLUS Real Time Clock, I/O Bus Interface, Two (2) Serial Ports, DMA, and the provision for 8087. \$795.00

C01668 - includes 68000, 256K RAM, Memory Expansion Bus, Z80 Interface, CP/M-68K, C Compiler, CPM-68K RAM Disk, CPM-80 RAM Disk \$799.00

C01668X - includes all of C01668 PLUS Real Time Clock, I/O Bus Interface, DMA, Two (2) Serial Ports, and the provision for up to four (4) 16081 Math Co-Processors. \$995.00

OPTIONS

Desktop Enclosure w/ power supply	\$125.00
Memory Expansion - 256K	\$467.00
Memory Expansion - 512K	\$659.00
I/O Module - IBM Compatible 4 slot (multiple I/O Modules allowed)	\$499.00
Math Co-Processor 8087	Call
Math Co-Processor 16081	Call
CPM-86	\$150.00

For more information:
see your favorite Dealer or contact:
HSC, INC.
262 East Main Street
Frankfort, NY 13340
1-315-895-7426
Reseller, and OEM inquires invited.

Circle no. 27 on reader service card.

for the external subroutine between the runtime library and the rest of the program. I could find no way to get such a subroutine into memory at the address specified while the Turbo package was running. Apparently the only way to do so is to compile the program to a "COM" or "CHN" file, then — using the system debugger — overlay the external routine at the appropriate address in the code file and save the altered program. I attempted this with a simple external lowercase-to-uppercase conversion function and it worked. The process is probably a little complicated for a novice programmer.

Conclusions

The system is easy to install. This phase of getting a new program to work often frustrates users, but Turbo Pascal handles the process automatically if you have one of the approximately 30 terminals that it knows about. If you don't, a detailed installation procedure is described in an appendix of the users' manual, allowing installation on any modern CRT terminal. I tried both methods and had no problems.

As initially installed, the command set for the editor is very similar to a sub-

set of WordStar. If you wish, you can change these commands.

Once installation is complete, using the system is a real pleasure. The integration of the compiler with the full-screen editor makes program development a snap. The compiler is incredibly quick, especially when translating a program for immediate execution in memory. The degree of interaction is very nearly that of working with a BASIC interpreter. If a compilation error occurs, the system will take you to the point in the program text where the error was detected. If an error occurs later, while running a program that was compiled in memory, the system will again take you to the source statement that caused the error. If an error is detected while running a "COM" or "CMD" file compiled by Turbo, it is also possible to find the source statement causing the problem.

Another important feature of Turbo Pascal is its size. It is small, both in terms of disk space and memory space. Although I did not try this, it appears that it would be possible to develop and run useful programs on computers with RAM memory as small as 32K.

The compiler accepts almost all standard Pascal statements. It is therefore possible to write highly portable pro-

grams. I had little trouble transferring programs between Turbo Pascal and UCSD Pascal. In many cases, no changes were required to compile and run the programs on either system. When changes were necessary, they were usually simple and straightforward. In most instances, the quality of the code produced was good, comparing favorably in size and speed with other Pascal and C compilers that I use on my system.

Turbo Pascal also has a large number of available extensions. Some allow easy access to low-level hardware and software functions of the host computer. Others include groups of built-in routines that are useful in many programs. Programmers often create libraries containing routines similar to these anyway, such as ones allowing control of the video terminal screen, but the Turbo system has already implemented many such facilities. Programs that use these extensions will probably not be directly portable to non-Turbo environments, but they are convenient.

The system is not without a few shortcomings, however. The absence of Get, Put, and Dispose may prove clumsy for some programs. Turbo Pascal is not acceptable for those applications that absolutely require separate compilation and overlay facilities. The alterations needed to include such capabilities in Turbo Pascal would probably negate many of its more desirable qualities. Also, the MicroCalc program, included as a programming example, requires a little polishing, mostly updating comments to take into account changes made in the program sections they refer to. Finally, the reference manual really needs some more work to bring it up to the quality of the rest of the system.

In short, Turbo Pascal would be an excellent purchase for two groups of computer users: those who already know Pascal and those who don't. Programmers who already use Pascal will find it an exceptional program-development system for the vast majority of applications. Persons who want to learn Pascal should acquire a good textbook as well as the Turbo package. The highly interactive nature of Turbo Pascal will make it nearly painless to "learn by doing" by experimenting with examples in the textbook.

TOTAL CONTROL:

FORTH: FOR Z-80®, 8086, 68000, and IBM® PC

Complies with the New 83-Standard

**GRAPHICS • GAMES • COMMUNICATIONS • ROBOTICS
DATA ACQUISITION • PROCESS CONTROL**

• **FORTH** programs are instantly portable across the four most popular microprocessors.

• **FORTH** is interactive and conversational, but 20 times faster than BASIC.

• **FORTH** programs are highly structured, modular, easy to maintain.

• **FORTH** affords direct control over all interrupts, memory locations, and i/o ports.

• **FORTH** allows full access to DOS files and functions.

• **FORTH** application programs can be compiled into turnkey COM files and distributed with no license fee.

• **FORTH** Cross Compilers are available for ROM'ed or disk based applications on most microprocessors.

Trademarks: IBM, International Business Machines Corp.; CP/M, Digital Research Inc.; PC/Forth + and PC/GEN, Laboratory Microsystems, Inc.

FORTH Application Development Systems include interpreter/compiler with virtual memory management and multi-tasking, assembler, full screen editor, decompiler, utilities and 200 page manual. Standard random access files used for screen storage, extensions provided for access to all operating system functions.

Z-80 FORTH for CP/M® 2.2 or MP/M II, \$100.00; **8080 FORTH** for CP/M 2.2 or MP/M II, \$100.00; **8086 FORTH** for CP/M-86 or MS-DOS, \$100.00; **PC/FORTH** for PC-DOS, CP/M-86, or CCPM, \$100.00; **68000 FORTH** for CP/M-68K, \$250.00.

FORTH + Systems are 32 bit implementations that allow creation of programs as large as 1 megabyte. The entire memory address space of the 68000 or 8086/88 is supported directly.

PC FORTH + \$250.00
8086 FORTH + for CP/M-86 or MS-DOS \$250.00
68000 FORTH + for CP/M-68K \$400.00

Extension Packages available include: software floating point, cross compilers, INTEL 8087 support, AMD 9511 support, advanced color graphics, custom character sets, symbolic debugger, telecommunications, cross reference utility, B-tree file manager. Write for brochure.



Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to (213) 306-7412



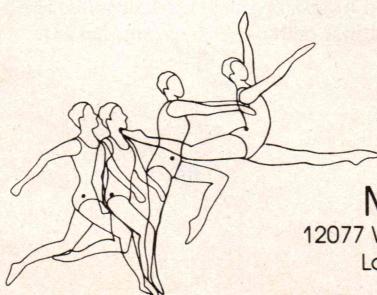
Circle no. 32 on reader service card.

MicroMotion

MasterFORTH

It's here — the next generation of MicroMotion Forth.

- Meets all provisions, extensions and experimental proposals of the FORTH-83 International Standard.
- Uses the host operating system file structure (APPLE DOS 3.3 & CP/M 2.x).
- Built-in micro-assembler with numeric local labels.
- A full screen editor is provided which includes 16 x 64 format, can push & pop more than one line, user definable controls, upper/lower case keyboard entry, A COPY utility moves screens within & between lines, line stack, redefinable control keys, and search & replace commands.
- Includes all file primitives described in Kernigan and Plauger's Software Tools.
- The input and output streams are fully redirectable.
- The editor, assembler and screen copy utilities are provided as relocatable object modules. They are brought into the dictionary on demand and may be released with a single command.
- Many key nucleus commands are vectored. Error handling, number parsing, keyboard translation and so on can be redefined as needed by user programs. They are automatically returned to their previous definitions when the program is forgotten.
- The string-handling package is the finest and most complete available.
- A listing of the nucleus is provided as part of the documentation.
- The language implementation exactly matches the one described in FORTH TOOLS, by Anderson & Tracy. This 200 page tutorial and reference manual is included with MasterFORTH.
- Floating Point & HIRES options available.
- Available for APPLE II/II+/III & CP/M 2.x users.
- MasterFORTH - \$100.00. FP & HIRES - \$40.00 each
- Publications
 - FORTH TOOLS - \$20.00
 - 83 International Standard - \$15.00
 - FORTH-83 Source Listing 6502, 8080, 8086 - \$20.00 each.



Contact:

MicroMotion

12077 Wilshire Blvd., Ste. 506
Los Angeles, CA 90025
(213) 821-4340

Circle no. 38 on reader service card.

A Professional Quality Z80/8080 Disassembler

REVAS Version 3

Uses either ZILOG or 8080 mnemonics
Includes UNDOCUMENTED Z80 opcodes
Handles both BYTE (DB) & WORD (DW) data
Disassembles object code up to 64k long!
Lets you insert COMMENTS in the disassembly!

A powerful command set gives you:

INTERACTIVE disassembly
Command Strings & Macros
On-line Help
Calculations in ANY Number Base!
Flexible file and I/O control
All the functions of REVAS V2.5

REVAS:

Is fully supported with low cost user updates
Runs in a Z80 CPU under CP/M*

Is normally supplied on SSSD 8" diskette

Revas V 3...\$90.00 Manual only...\$15.00

California Residents add 6 1/2% sales tax

REVASCO

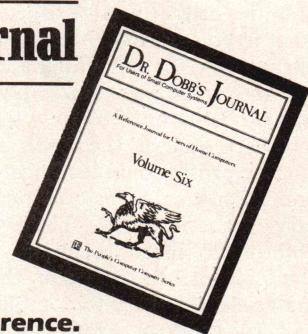
6032 Charlton Ave., Los Angeles, CA. 90056
(213) 649-3575

*CP/M is a Trademark of Digital Research, Inc.

Circle no. 61 on reader service card.

Dr. Dobb's Journal

Bound Volumes



**Every Issue Available
For Your Personal Reference.**

We are pleased to offer this special discounted price to **DDJ** readers who order directly from us. From the nostalgia of Volume One—with authors like Steve Wozniak, Dennis Allison, Sol Libes, and more—to the technical maturity of Volume Six, **Dr. Dobb's Journal Bound Volumes** are the ideal addition to your reference collection. They contain many issues which are no longer available and you get twelve issues for the price of seven individual back issues!

Send \$23.75 each for volumes 1-5, \$27.75 for volume 6, or \$125 for all 6 and SAVE!

Please add the following per book: \$2.50 for UPS, \$1.25 for U.S. Mail, or \$2.00 for Foreign postage. Delivery times are one week for UPS or 6-10 weeks for U.S. or Foreign Mail.

**Mail To: Dr. Dobb's Journal
2464 Embarcadero Way, Palo Alto, CA 94303**

Circle no. 78 on reader service card.

BOOK REVIEWS

High-Tech Consulting

By John Zarella

Published by Microcomputer Applications, November 1983

\$18.95 paper, 167 pages

Reviewed by Robert Clark

Consulting is one of the most attractive and potentially lucrative ways for a hardware or software professional to establish an independent enterprise in the field of computer systems. A consulting business can be started on a shoestring, and can be initiated and run effectively by a single person. The business may begin returning profits almost at once, as high-tech companies will pay handsomely for the expertise needed to design new products using the latest technologies.

For those eager to capitalize on the opportunities presented by becoming a consultant, Zarella has some sound if sobering, advice on the business aspects of consulting. There are at least as many pitfalls as there are gold mines in this field, and anyone seriously considering starting out on his or her own will appreciate the pointers offered in *High-Tech Consulting*. Zarella spends little time recounting the attractive aspects of consulting; if you finish this book with as much enthusiasm for becoming a consultant as you had when you started it, your business will probably be extremely sucessful.

Part one of the book starts by assuming professional technical competence and describing other qualifications necessary for consulting. These include the abilities to listen to clients and analyze their needs, to estimate the time and materials necessary to complete a project, and to produce proposals and provide progress reports and other documentation on the work being done. Verbal communication skills are also needed to convince potential clients that you are the person for the job, and to ensure that they understand your approach to the problem. The latter includes conducting design reviews and explaining your work to technical and non-technical clients.

Zarella goes on to examine several notions about the consulting lifestyle. He admits that there are some advantages to being your own boss, selecting the jobs

on which you will work, setting your rates, and managing your business and personal finances. He is quick to point out, however, that your source of income is always at risk, that your periods of free time will be unpredictable, and that your credit rating will probably disappear.

The second part of the book deals with setting up a consulting business. Such a business may be anything from an after-hours income supplement to a corporation with sizeable office space and many employees. The book is geared toward the one-person enterprise, and briefly covers such things as estimating startup costs, writing a business plan, obtaining a business license, paying taxes, recordkeeping, finding part-time or full-time employees, and dealing with Government red tape. (Think twice before taking a Government contract; filling out forms in triplicate can wipe out profit margins.)

In part three, the day-to-day aspects of running a consulting business are discussed. As a consultant, you will find that substantial energy will go into marketing your abilities, meeting with prospective clients, writing proposals, and other non-billable activities. Setting your rates and bidding on fixed-price jobs must be done with this in mind. These topics are covered along with suggestions on keeping technical records.

Zarella finishes up in part four with the topic of protecting yourself, your family, and your business with contracts, collection schedules, legal advice, patents and copyrights, and insurance, an Appendix of references and selected reading provides additional "how to" sources for the topics covered in this book.

If you follow all of the suggestions in *High-Tech Consulting* for establishing and maintaining your enterprise, you will have a model small business. You will also find yourself swamped in the details of running it. This book is valuable in that it offers advice on virtually every business aspect of consulting, but a one-person business can selectively disregard some of these. For instance, everyone should know what they are committing to when they sign a contract or non-disclosure agreement, but not everyone needs a business plan. Too much paperwork can make one lose sight of the fact that high-tech consulting is the business of hardware and software development, and a successful consultant should be able to enjoy doing just that if sound business practices are followed.

How to Create Your

Own Computer Bulletin Board

By Lary L. Myers

Published by Tab Books Inc.

\$12.50 (paper), 214 pages

Reviewed by James Moran*

Although no new ground is covered in *How to Create Your Own Computer Bulletin Board*, Lary Myers has done an admirable job of compiling program code for this book.

Designer of the FORUM80 and TABLOID BBS (Bulletin Board System) in Albany, New York, Myers has written a book that will be particularly useful to computer clubs that are interested in setting up their own boards. The first few chapters of the book serve a dual purpose in that they provide a technical introduction to the programs that follow as well as suggest some preferred BBS designs.

Although a little light on the narrative side, the book covers the basics of bulletin board design and does a good job of pointing out the danger areas in running these boards. In particular, serious BBS designers will appreciate the section on board security and integrity.

The majority of the book is taken up by BASIC and assembler programs that fully document a functional board for Apple and TRS-80 users. With the notable exception of the LTERM intelligent terminal program for TRS users, all of the assembler listings are nicely commented and should prove simple to modify.

The only serious problem with this book is the print quality of some of the listings. Whether this was due to the dot matrix printer that the author used to reproduce the listings or to a quality control problem at the printers is difficult to determine. In any case, for those who are interested, the publisher offers to supply the BBS programs on disk for \$30. In view of the fact that the book is in large part a 160-page compendium of program listings, those BBS enthusiasts who are considering the implementation of their own board might want to bypass the book and just order the program diskette.

*Copyright © 1984 CompuSyn.

Announcing *Dr. Dobb's*

Fifth-Generation Programming Competition

The Challenge: To write a program that applies artificial intelligence techniques to a practical problem, and to extend the present boundaries of what a microcomputer can do.

No rigorous definition of the field of artificial intelligence exists, but there are some programming problem domains that are commonly referred to as AI. These include, among others, pattern recognition, learning, logical reasoning, the representation of knowledge, planning and problem solving. Programs that make a stab at dealing with natural language or with speech or with visual input are regarded as AI programs, as are automatic-programming tools and chess programs.

AI's earliest successes were in game playing, but recent work in the area of expert systems has shown that artificial intelligence techniques can be of real use in practical real-world problem areas like medical diagnosis, geological research and chemical identification. Perhaps because of the success of expert systems, which typically require the processing of large amounts of information, the belief is widespread that microcomputers are inadequate for AI work. We disagree. We believe that there are AI problems waiting to be solved via microcomputer, and we conceive this competition to challenge the brightest microcomputer programmers we know of — *Dr. Dobb's* readers — to find and solve some of those problems. A fantasy? Perhaps. But *Dr. Dobb's Journal* has been publishing Realizable Fantasies since 1976.

The Rules: The following items must be included in each entry: a functioning program on disk, a well-documented listing of the program, a prose description of the program explaining what it does and how it does it, and the entrant's name and address. The program must be written in a high-level language for a microcomputer, and the entry must be postmarked no later than September 30, 1984 and received by October 30, 1984 (*overseas entrants take note*).

Although your choice of specific language, hardware, problem domain and approach to the problem are all open, we offer some guidelines as to how we will be judging the entries. Your program will be judged for originality and significance of contribution first, and second on its intrinsic merits as a program: modularity of design, efficiency, portability, clarity, and so on. A program that executes in 64K of memory will get higher marks than an equivalent program that requires 96K, but a 128K program that does significantly more may rate higher than either. If your choice of hardware or disk format is an unusual one, you might query us before submitting your entry.

The Payoff: The winner will receive a \$1000 prize and will have the program and prose description published in *Dr. Dobb's*. In the tradition of *Dr. Dobb's*, this contribution to the advance of microcomputer software is to be placed in the public domain with full credit to the author. Support for the concept of public domain software is part of *Dr. Dobb's* heritage, but it may not be part of yours. This competition is for those who genuinely want to make a contribution to the advance of microcomputer software. Send your entry to:

Dr. Dobb's Journal, 2464 Embarcadero Way, Palo Alto, CA 94303.

16-BIT SOFTWARE TOOLBOX

by Ray Duncan

Professional Basic

Neil Bennett's outstanding BASIC implementation for the IBM PC, called "Professional Basic," is now available through Morgan Computing Company, Inc. (10400 North Central Expressway, Suite 210, Dallas, TX 75231). This version of the language sets new standards with its fantastic window-oriented debugging tools, dynamic syntax checking, use of the Intel 8087 numeric co-processor for fast floating-point operations, and support for the 8086/88 CPU's full 1-megabyte address space for both programs and data. The user interface has a distinctive, elegant style that is unbeatable for program development. For more details, read the review in the April 1984 issue of *Byte* magazine. If you own an IBM PC, and must program in BASIC for whatever reason, you owe it to yourself to pick up the phone and order a copy of Professional Basic. Users of other machines take heart; versions are being contemplated for most of the popular 16-bit microcomputers including 68000-based machines.

The Filepath Utility

"Filepath," a small but indispensable utility from SDA Associates (P.O. Box 36152, San Jose, CA 95158), has changed my life for the better. Written by Bernie Belew and selling for the minuscule fee of \$12.95 plus tax, Filepath does for data files what the MS-DOS "Path" command does for executable program files. This is extremely helpful when using programs that refer to the disk for overlays, messages, and the like. For example, you can now put one copy of WordStar and its overlay files into a single subdirectory on your hard disk, point to that subdirectory with "Path=" and "Filepath=" specifications in your AUTOEXEC.BAT file, and use the word processor successfully from anywhere in the directory structure.

Beating the Laws of Thermodynamics

Those of you with a taste for computing esoterica might like to read the article "Computing without Dissipating Energy," in *Science* (March 16, 1984, page 1164). When I was a chemistry major, the Three Laws of Thermodynamics were taught to us as: (1) You can't get something for nothing, (2) You can't break even until you get to absolute zero, and (3) You can't get to absolute zero. There is now a heated (if you'll excuse the ex-

pression) debate going on between one group of physicists who believe computation inherently requires the expenditure of energy and some scientists who think the energy dissipation per logical operation could be brought arbitrarily close to zero with radically different technology. If you're interested, brush up on your Turing machine theory and lose yourself within the pages of *Science*.

Floating-Point Benchmarks

Results continue to flow in on the Bill Savage floating-point benchmark recently published in this column. In spite of its deficiencies, this benchmark seems to have sparked an incredible amount of reader interest. Versions for two of the "missing" languages, LISP and Logo, have been contributed and may be found in Listings One and Two (page 83). I plan to publish a corrected and expanded list of timings and errors within the next three months.

Mark Seage of Lawrence Livermore Labs came through with results for the Cray I and Cray X-MP processors and certainly made me eat my words about the performance of 8086/8087-based microcomputers compared to mainframes. The timing for the latter, incidentally, was an incredible 0.00012 seconds with an absolute error of 1.7E-10. He notes that this particular benchmark cannot be vectorized, but that in a benchmark that was vectorizable an additional speedup of 10 to 100 times could be expected. Now that I am confronted with an example of the number-crunching performance of a *real* mainframe that I can relate to, I must admit to being a bit awed.

At the other end of the computer spectrum, Jim Benenson wrote to defend the reputation of the Sinclair ZX-81. This machine's BASIC was previously reported to complete the Savage benchmark in about 1 day. Jim observed an execution time of 935 seconds in "fast mode" and an absolute error of 3.0E-1.

An error occurred in my transcription of the result for WS Systems Forth running on the 8086 with 8087 support. The actual error was 2E-13, not 1E-3 as printed in the column.

In the course of their comments on the original benchmark program, several readers have recommended *Software Manual for the Elementary Functions*, by William Cody Jr. and William Waite (Prentice Hall Inc., 1980). The book sells

for about \$20, and is "designed to help a programmer without a degree in numerical analysis implement the elementary mathematical functions," says Delbert Franz. It gives approximations for a variety of precisions and includes detailed flow charts and notes for a variety of floating-point bases ... as well as for fixed-point machines."

More on the Microsoft 8086 Assembler

My earlier column on some "anomalies" in the Microsoft Macro Assembler provoked a few indignant letters and some reports of further bugs. Several people came to the defense of the Assembler, saying that some of the problems that I considered bugs were simply artifacts of its internal storage format for variables. One person even stated that the Microsoft Assembler's display of word values in the opposite byte order from the actual physical storage in memory was "not a bug, but a feature!"

Sorry folks, I just can't agree. When you are using an assembler, you are working at the machine level and can expect to be staring at plenty of object code dumps while debugging. If the listing doesn't correspond *exactly* to what's in memory, it's only adding to your mental workload and creating unnecessary opportunities for additional errors. The same argument applies to the representation of the actual binary value of equates. If Microsoft wants to promulgate a philosophy of writing assemblers that is different from everyone else in the world, it could at least warn us in the manual.

Dan Rollins, author of an excellent assembly language column in *PC Age* magazine, writes, "There is another very nasty bug [in the Microsoft Assembler] that can cost hours of debugging. That is, when the symbol in

ADD reg/mem16,offset symbol

has a segment offset of less than 128, the assembler generates the sign-extended version of the addressing mode byte. This is very considerate — it saves an entire byte in the executable module. However, the reference that's generated is not relocatable. Therefore if the position of the specified data changes during the link process, you end up with a logic error that's not your fault. The same problem exists for SUB, AND, CMP, OR, etc., when the address operand just happens to be early in the module.

"Here's a handy hint for your readers

Sometimes when you link a bunch of modules, you want to be sure that their segments will be loaded into memory in a specific order. For instance, you may want your DATA_SEG to come after your CODE_SEG. Alas, the physical placement of the segment in the source code has no effect on the placement in memory. The undocumented way to do this is to name your segments so that they are alphabetically in the order that you want them in memory. But, this only applies to segments in a single module.

"The Linker documentation clearly states that the overriding priority of segment placement is the order in which the object modules are named in the LINK command line. You can let that foul you up, or you can use it to your advantage.

"First, create several OBJ modules that each define an empty segment. For instance, one module would look like this:

```
; ** dummy module DATA_MOD.OBJ
data_seg segment public
data_seg ends
end
```

"DATA_SEG would be the same name as a segment in the module you

want to reorganize. Then have the linker load these 'dummy modules' before loading the real code:

```
LINK code_mod +stack_mod
+data_mod +myprog;
```

"Rearranging the order of the dummy modules also rearranges the placement of the modules in memory. Verify it by reading the link map.

This technique is most useful when using the GROUP pseudo-op for creating COM format files. Often you'll want to define your data at the top of a program in order to avoid the phase errors and 'Operand must have size' errors that can happen when you have forward references in your code."

Page 5-40 of the Microsoft Assembler manual alludes to another way of controlling the placement of a segment. The "MEMORY" combine-type entry on a segment declaration is supposed to force that segment to be loaded at a higher address than all other segments being linked together. However, it doesn't work as advertised — just another "feature," I guess.

One particularly amusing report came

from a user who found that mistyping "para" as "par" in a segment declaration can cause the Microsoft Assembler to crash without any warning or error message. The mechanism by which this could be occurring is hard to imagine. After all, the source file is, from the assembler's point of view, simply a data structure to be transformed, right? It's a good thing our word processors don't crash every time we misspell a word!

Toolbox Installment

Listing Three (page 84) contains a set of three interdependent subroutines to convert a word or byte into the equivalent hexadecimal ASCII representation. The routines are written in a form suitable for the Microsoft 8086 Assembler, though the code should also be acceptable with little or no change to Digital Research or Intel assemblers.

BBJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 198.

16-Bit Listing One

Logo version of the Savage benchmark, adapted from program contributed by Tom Prince.

```
to savage ;Savage benchmark in LOGO
make "a 1
repeat 2500 [make "a (tan atan exp ln sqrt :a*:a 1) +1]
;note that LOGO atan is like FORTRAN atan2
(print [a = [ :a)
end

to tan :x
output (sin :x)/(cos :x)
end
```

End Listing One

Listing Two

LISP version of the Savage benchmark, contributed by Morton Goldberg.

```
~  
~ An IQ LIST version of Savage's Benchmark Program  
~  
(DEF 'BMRK  
' [LAMBDA (N)  
  (OR N (SETQ N 2500))  
  [PROG (A)  
    [SETQ A 1.0D)
```

(Continued on next page)

16-Bit (Listing Continued, text begins on page 82)

Listing Two

```
LOOP
  (SETQ A (+ 1.0D
    (TAN (ATAN (EXP (LN (SQRT (* A A ) T) T) T) T) T)))
  (SETQ N (SUB1 N))
  (COND [(LE N 0) (RETURN (SUB1 A))]
  (GO LOOP))]
```

End Listing Two

Listing Three

```
; Intel 8086 subroutines to convert a binary
; word or byte to hexadecimal ASCII format.
; Ray Duncan, March 1984.

word_to_hex proc near
  ;convert 16-bit binary word
  ; to hex ASCII
  ;call with AX=binary value
  ;           DI=addr to store string
  ;           (4 characters)
  ;returns AX, DI, CX destroyed
  push  ax
  mov   al,ah
  call  byte_to_hex ;convert upper byte
  pop   ax
  call  byte_to_hex ;convert lower byte
  ret
word_to_hex endp

byte_to_hex proc    near
  ;convert binary byte to hex ASCII
  ;call with AL=binary value
  ;           DI=addr to store string
  ;           (2 characters)
  ;returns AX, DI, CX destroyed
  sub   ah,ah
  mov   cl,16
  div   cl
  call  ascii
  stosb
  mov   al,ah
  call  ascii
  stosb
  ret
byte_to_hex endp

ascii  proc    near
  ;convert bottom 4 bits in AL
  ;into the hex ASCII character
  add   al,'0'
  cmp   al,'9'
  jle   ascii2
  add   al,'A'-'9'-1
  ascii2: ret
  ascii  endp
```

End Listings

GGM — FORTH™ has HELP*
for Z80¹ using CP/M²

GGM—FORTH, a complete software system for real-time measurement and control, runs on any Z80 computer under CP/M using an extended fig-FORTH vocabulary.

GGM—FORTH features:

- Open multiple CP/M files, in any combination of direct-access and sequential-access, fully compatible with all CP/M utilities
- Char. in/out uses CP/M console, lister, file, or port
- On-line HELP* provides instant access to definitions in the run-time GGM—FORTH dictionary
- HELP* file is easily extended to include user definitions using HELP* utility
- HELP* is available during full-screen editing

Complete system and manuals \$150.
Manuals only: \$20.
Introductory System: \$35.

GGM SYSTEMS, INC. (617) 662-0550
135 Summer Ave., Reading, MA 01867

¹Z80 is a trademark of Zilog, Inc.

²CP/M is a trademark of Digital Research, Inc.

Circle no. 25 on reader service card.

THE PROGRAMMER'S SHOP

helps compare, evaluate, find products. Straight answers for serious programmers.

SERVICES

- Programmer's Referral List
- Dealer's Inquire
- Compare Products
- Newsletter
- Help find a Publisher
- Rush Order
- Evaluation Literature free
- Over 300 products
- BULLETIN BOARD - 7 PM to 7 AM 617-461-0174

"C" LANGUAGE LIST OUR PRICE PRICE

	EDITORS	Programming
APPLE: AZTEC C - Full, ASM	\$199	call
8080: BDSC C - Fast, popular	150	125
8080: AZTEC C - Full	199	call
Z80: ECOSOFT - Fast, Full	250	225
8086: C86 - optimizer, Meg	395	call
8086: Lattice - New 1.1 & 2.0	500	call
Microsoft (Lattice) MSDOS	500	call
Digital Research - Megabyt 8086	350	269
Desmet by CWare - Fast	8086	109
	109	99

BASIC ENVIRONMENT

Active Trace - debug	8080/86	\$ 80	72
MBASIC-80 - MicroSoft	8080	375	255
BASCOM-86 - MicroSoft	8085	395	279
CB-86 - DRI	CPM86	600	439
Prof. BASIC Compiler	PCDOS	345	325
BASIC Dev't System	PCDOS	79	72

FEATURES

C INTERPRETERS for MSDOS - Ask about one for beginners for \$85 or full development for \$500.

C HELPER includes source in C for MSDOS or CPM80 for a DIFF, GREP, Flow-charter, C Beautifier and others. Manage your source code easier. \$125.

PROLOG86 Interpreter for MSDOS includes tutorials, reference and good examples. Learn in first few hours. For Prototyping, Natural Language or AI. \$125.

Our Free Report: PRODUCTIVITY - MSDOS

Assume use of compiler and typical editor. What commercial or public domain products, what techniques improve productivity? "Productivity with MSDOS" is a growing document with some answers. Call to request it. Help improve it. Earn \$50 credit toward any purchase when we add any description, code, or idea received from you.

RECENT DISCOVERIES

PROFILER - Examine MSDOS program execution speeds. Determine where to improve programs in any Microsoft language, Lattice, or C86. Make histograms that show time spent in portions of your program, and doing MSDOS I/O, etc. \$175.

EDITORS Programming

C Screen with source	8080/86	NA	60
EDIX - clean	PCDOS	195	149
FINAL WORD - for manuals	8080/86	300	215
MINCE - like EMACS	CPM, PCDOS	175	149
PMATE - powerful	CPM	195	175
	8086	225	195
VEDIT - full, liked	CPM, PCDOS	150	119
	8086	200	159

FORTRAN

MS FORTRAN-86 - Meg	MSDOS	\$350	\$255
SS FORTRAN - 86	CPM-86	425	345
FORTRAN-80 - 66 decent	CPM-80	500	350
INTEL FORTRAN - 86	IBM PC	NA	1400
DR FORTRAN COMING			
RM FORTRAN COMING			

LANGUAGE LIBRARIES

C to dBASE interface	8080/85	\$125	\$115
C Tools 1 - String, Screen	PCDOS	NA	115
C Tools 2 - OS Interface	PCDOS	NA	92
C Tools-XOR - Graphics, Screen	PCDOS	NA	95
FLOAT 87 - Lattice, PL1	PCDOS	NA	115
GRAPHICS: GSX - 80	CPM80	NA	75
HALO - fast, full	PCDOS	200	175
Greenleaf for C - full	PCDOS	NA	165
ISAM: Access Manager - 86	8086	400	300
BTRIEVE - many languages	PCDOS	245	215
PHACT - with C	PCDOS	NA	250
FABS	CPM80	150	135
PASCAL TOOLS - Blaise	PCDOS	NA	115
SCREEN: Display Mgr. 86	8086	500	375
PANEL-86 - many languages	PCDOS	350	315
WINDOWS for C	PCDOS	NA	115
Virtual Screen - Amber	PCDOS	295	call

PASCAL

ENVIRONMENT	LIST	OUR	
PASCAL MT + 86	CPM86/IBM	\$400	\$289
without SPP	CPM80	350	249
MS PASCAL 86	MSDOS	350	315
SBB PASCAL - great, fast	PCDOS	350	315
PASCAL 64 - nearly full	COM 64	99	89
SBB Jr - best to learn	PCDOS	NA	95

OTHER PRODUCTS

AKA ALIAS - improve DOS	PCDOS	NA	60
Assembler & Tools - DRI	8086	200	159
COBOL - Level II	8086	1600	1275
CODESMITH-86 - debug	PCDOS	149	139
IQ LISP - full 1000K RAM	PCDOS	175	call
Janus ADA - solid value	PCDOS	500	449
MBP Cobol-86 - fast	8086	750	695
Microshell improve CPM	8080	150	125
Microsoft MASM-86	MSDOS	100	85
PL/1-86	8086	750	560
PLINK-86 - overlays	8086	350	315
POWER - recover files	8080/86	169	139
MicroPROLOG	PCDOS	NA	265
READ CPM86 from PCDOS	PCDOS	NA	55
READ PCDOS on an IBM PC	CPM86	NA	55

Note: All prices subject to change without notice. Mention this ad. Some prices are specials.

Ask about COD and POs.
All formats available.

Call for a catalog, literature, and answers

800-421-8006

THE PROGRAMMER'S SHOP™

128-D Rockland St., Hanover, MA 02339.
Visa 617-826-7531, Mass: 800-442-8070 MasterCard

Circle no. 49 on reader service card.

C Programmers: Program three times faster with Instant-C™

Instant-C™ makes programming three or more times faster by eliminating the time wasted by traditional compilers. Many repetitive programming tasks are automated to make programming less frustrating and tedious.

- Two seconds elapsed time from completion of editing to execution.
- Full-screen editor integrated with compiler; compile errors set cursor to trouble spot.
- Editor available any time during session.
- Symbolic debugging; single step by statement.
- Automatic recompilation when needed. Never a mismatch between source and object code.
- Directly generates .COM, .EXE, or .CMD files.
- Follows K & R—works with existing source.
- Single, integrated package.
- Works under PC-DOS*, MS-DOS*, CP/M-86*.

More productivity, less frustration, better programs. Instant-C™ is \$500. Call or write for more information.

Rational
Systems, Inc.

(617) 653-6194
P.O. Box 480
Natick, Mass. 01760

*Trademarks: PC-DOS (IBM), MS-DOS (Microsoft), CP/M-86 (Digital Research, Inc.)
Instant-C (Rational/Systems, Inc.)

Circle no. 58 on reader service card.

C/UNIX PROGRAMMER'S NOTEBOOK

by Anthony Skjellum

In previous columns, I have alluded to 8086-family C compilers that support large-memory models. Before discussing some of the compilers that provide such features, I thought it worthwhile to discuss the memory model concepts of the 8086 and how these concepts impact C compilers implemented for this microprocessor family. I will discuss the advantages and drawbacks of several memory models used by existing C compilers, and present code to help overcome some of the limitations of small-memory model compilers.

For those readers who are not interested in the details of 8086 C compilers, large-memory models, or long pointers, there is still some interesting material in this column. Specifically, several routines that are included here illustrate real-life code that will interface C and assembly language. Most compiler manuals are terse on this subject, so some actual code may help drive home the concepts involved.

Background

Before plunging into a discussion of memory models, a brief introduction to the 8086 architecture is necessary. This material will help to illustrate why different compilers use different addressing schemes.

The 8086/88 microprocessors support 20-bit addressing. This fact allows the microprocessor to address in excess of 1 million bytes. All the registers are 16 bits wide, however. This implies that some segmentation scheme must be used in order to address more than 64K of memory. The technique used involves four 16-bit segment registers: CS, DS, ES, and SS. These registers are the code-segment, data-segment, extra-segment, and stack-segment registers, respectively. Depending on the instruction used, different segment registers come into play in determining the complete 20-bit address. In forming a complete address, the segment address is always shifted left four bits. Note that a segment register by itself addresses memory on 16-byte boundaries. Sixteen-byte regions addressable by segment registers are known as paragraphs. Although paragraphs and paragraph alignment are not normally of interest to C programmers, they are sometimes important when developing assembly-language interface code for C.

When discussing long pointers, a special notation is used. Because the address is split, it is written in the form:

segment:byte_pointer

where *segment* is the segment, and *byte_pointer* is the 16-bit, low-order part of the address. A typical example of such an address would be "es:bx," which means "segment specified by es register and offset from this segment specified by the bx register." This notation is used throughout the listings included with this column.

Machine instructions often differentiate between inter- and intrasegment operations. For example, there are "near" and "far" CALL instructions.

I will now outline several memory models.

8080 Memory Model

The 8080 memory model is just what its name implies. All segment registers are set equal, so that only a total of 64K is available for a program. This model is seen mostly under CP/M-86 but is occasionally used by MS-DOS programs. None of the C compilers that I have seen restrict programs to this model.

Small-Memory Model

Many programs can work comfortably with only 64K of data space and 64K of program space. Such a model results when the CS and DS registers are set to different blocks of memory (up to 64K each). Normally, ES and SS are set equal to DS, so that all data and stack memory resides in the same block of memory. This model is fine as long as programs and data requirements are small enough to fit within the 64K limits. Most C compilers support only this model.

```
typedef union __lptr
{
    long    _llong;      /* long format */
    char    _lstr[4];    /* character format */
    WORD    _lword;      /* long-word format */
} LPTR;
```

Figure 1.

```
typedef struct __lword
{
    unsigned _addr;      /* address */
    unsigned _segm;      /* segment */
} WORD;
```

Figure 2.

Large-Memory Model

In a large-memory model, all addresses refer to the full 20-bit range. All subroutine calls are "far" calls, and all data is referred to with long pointers. Long pointers include a segment- and byte-address pointer (thus occupying 32 bits). Only a few C compilers support this model. The reason that most compilers don't support this model is its complex code generation. I will mention more on this later.

Hybrid

A useful hybrid of the small- and the large-memory models is the one where only 64K of program space is provided but long pointers for data are used. This model offers speed advantages for programs that require more data storage but are moderately small.

One other possibility would be a large-code/small-data model, which would be used for programs with small data requirements but large code requirements.

One type of model that has not been considered is one that supports a large stack. A large stack would support more than 64K of items. Implementing this feature would slow program execution significantly, because stack references would be complicated.

Which Model Is Better?

As long as a C program can fit within the small-memory model, there is a distinct speed advantage in using this model. The large-memory model produces longer (and somewhat slower) programs because of the greater generality of each instruc-

tion produced (ability to refer to 1024K instead of 64K of memory requires longer pointers and more checks). Because the 8086 doesn't provide many instructions to manipulate the long pointers, many additional instructions must be generated for pointer-related operations (which also include all memory references).

Specific examples of the lack of 8086 instructions involve incrementing and decrementing long pointers. Note that a long pointer is not just a 32-bit word. The upper 16 bits is a segment address that must be treated accordingly when crossing 64K boundaries. Examples of implementing these features in software are included in Listing Five (11int.asm: examples: 1inc and 1dec functions, page 96).

Thus, both models have drawbacks. Speed is gained at the expense of (essentially) unlimited program and data space. Use the large-memory model for big programs that use big chunks of data. Otherwise, stick with the small model.

Drawbacks of the Small-Memory Model

Assuming that you use the small-memory model (by choice or because of your compiler), everything will run smoothly until it becomes necessary to deal with memory outside of the C data-address space. For example, it might be nice to use large buffers for copying files or for keeping help information. Another possibility would involve accessing special locations in the memory map.

The ability to use long pointers in a small-memory model can be implemented with relative ease. A set of such routines is presented in Listings One through Five (pages 88-104). A description of the Long Pointer Package and applications for the package form the remainder of this column.

The Long Pointer Package

The Long Pointer Package supplements a C environment by allowing references to memory locations to occur anywhere in the 20-bit address map. This is done by defining a new data type, LPTR (via a `typedef`), as shown in Figure 1 (page 86) where LWORD is defined as the structure shown in Figure 2 (page 86). This format for LPTR makes the addresses defined directly compatible with normal long pointers used at the assembly level. These long pointers are stored in the 8080 style: least-significant byte of address first, most-significant byte of segment last.

The lowest-level routines that support long memory references are, of necessity, coded in assembly language. The routines that implement many of the lowest-level functions in a noncompiler-specific way are included in Listing Four (11sup.asm). Routines that implement functions for Aztec C86 (a typical 8086 C compiler) Version 1.05i are included in Listing Five

(11int.asm). The user may have to modify these routines to work with other C compilers if register usage or stack arrangements differ.

In order to actually use the routines with C programs, the header file "1sup.h" must be included at the beginning of modules that use or refer to LPTR data types. The "1sup.h" file refers to "1sup.h" also. These two headers are presented in Listings Two and Three, respectively.

Supported Functions

The package supports a number of functions involving long pointers. There are routines to add offsets to long pointers and to copy memory between long pointers and routines to return data addressed by long pointers. A complete list of these functions is included in Table I (below). In this table, the file in which the function is located is mentioned.

An Example

One useful application of long pointers under MS-DOS 2.0 involves accessing a program's environment block. The environment block is a Unix-like set of environment variables and values. This is normally used to affect some particular aspects of program execution. Specifics about the environment address are included in the inset on page 88. Interested readers should also refer to the DOS 2.0 users' manual for more details.

The example program env.c reads the environment block and displays the contents of the whole block on the console. In effect, it provides the same listing feature as the MS-DOS SET command.

Conclusion

In this column I have discussed various aspects of memory models for 8086 C compilers. I have included a set

file: 1sup.c (some C support routines)

lassign(dest,source)	assign long pointers
lstrcpy(dest,source)	long string copy
lprint(lptr)	debugging routine for printing LPTRs

file: 11int.asm (Aztec C dependent support routines)

lptr(lptr,sptr)	form a long pointer from a normal short C (ds relative) pointer
lchr(lptr)	return character addressed by long pointer
lint(lptr)	returns int/unsigned addressed by long pointer
l_stchr(lptr,chr)	stores char at location lptr.
l_stint(lptr,intgr)	stores int at location lptr.
lload(dest,lptr,len)	general purpose copy to short pointer area (ds relative) from long pointer area
lstor(lptr,src,len)	reverse if lload()
linc(lptr)	increment long pointer
ldec(lptr)	decrement long pointer
ladd(lptr,offset)	add unsigned offset to lptr
lsub(lptr,offset)	subtract unsigned offset from lptr
lsum(lptr,offset)	add signed offset to lptr
lcopy(dest,src,len)	general purpose long to long copy (can copy up to 1024K of memory)

file: 11sup.asm (compiler independent functions)

linc	increment a long pointer
ldec	decrement a long pointer
ladd	add an unsigned offset to a long pointer
lsub	sub an unsigned offset from a long pointer
lsum	add a signed offset from a long pointer
lcopy	general copy routine

Table 1.

Environment Block Address

C compilers under MS-DOS normally produce .EXE files. For .EXE files, a program-segment prefix is created by DOS 2.0 and higher. The segment address of this prefix is es:0 when the user program begins. At offset 002CH from this address is stored the segment address of the environment table. Only a segment is stored: the offset from the segment is again zero. Thus, the contents of es:2CH is the address of the environment block.

Normally, C compilers have a maintenance routine that is given control at the start of program execution. In Aztec C86, this routine is called \$begin and is located in the calldos.asm module included with the compiler. The user must define an external variable in calldos.asm for the benefit of env.c for the segment address to be accessible as a long pointer. The procedure for this operation is detailed in the comments included in Listing Six (env.c, page 104).

Allocation of Memory

If a C program intends to use DOS memory allocation in conjunction with the long pointers, it must also be sure to shrink its memory allocation, using the MS-DOS SETBLOCK function. This is normally done in the initial maintenance routine of the C runtime system. In Aztec C, this must be done in \$begin.

of C and assembly-language functions that supports long pointers under a small-memory-model environment. With this package, users can enjoy the best of both worlds: access to arbitrary amounts and locations of memory, while retaining the efficiency of short pointers for regular code and pointer operations. For users of compilers that only support the small model, this package allows access to features that were previously off-limits to 8086 C programmers.

BBJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 199.

C/UNIX (Text begins on page 86)

Listing One

```
*****  
*  
*      lsup.c           created: 25-Mar-84  
*  
*      long pointer support for small memory model 8086 C compilers.  
*  
*      version 1.00 as of 25-Mar-84  
*  
*      Copyright 1984 (c) Anthony Skjellum.  
*      All rights reserved.  
*  
*      This program may be freely distributed for all non-commercial  
*      purposes, but may not be sold.  
*  
*      The routines contained here are designed to be portable to  
*      a large variety of compilers.  Currently they have been tested  
*      with Aztec C86 v 1.05i only.  
*  
*      Modules comprising this package:  
*  
*          lsup.c      this file.  
*          lsup.h      header/definition file.  
*          _lsup.h     lower level header for this file  
*          llsup.asm   assembly language support (compiler  
*                         independent)  
*          llint.asm   compiler interface code (compiler  
*                         dependent)  
*  
*      Subroutines included here:  
*      (those marked with an asterisk are only included if compiler  
*      used lacks some preprocessor support feature)  
*  
*****
```

```

#include "_lsup.h"                      /* header with definitions */

/*
   Special routines: Included only if compiler lacks one of
   several features.
*/

/* lassign(dest,source): assignment of type LPTR to the left */

#ifndef MSUBST

lassign(dest,source)
LPTR dest;
LPTR source;
{
    dest._llong = source._llong;      /* assignment */
}

#endif

/*
   General purpose routines:
*/

/* llstrcpy(dest,src): copy null terminated strings between long ptrs */

llstrcpy(dest,src)
LPTR *dest;
LPTR *src;
{
    char chr;                      /* temporary */
    while(1)                      /* loop */
    {
        chr = lchr(&src);        /* get a character */
        l_stchr(&dest,chr);      /* store a character */

        linc(&dest);            /* increment destination ptr */
        linc(&src);             /* and source pointer */

        if(!chr)                 /* we are done at eos */
            break;
    }
}

/* debugging routines: */

lprint(lptr)
LPTR *lptr;
{
    printf("%lx",lptr->_llong);
}

```

End Listing One

Listing Two

```

#include "_lsup.h"

/* place any special function specifications (defined in lsup.c) here: */

```

End Listing Two

(Listing Three begins on next page)

Listing Three

```
*****  
*  
*      _lsup.h          created: 25-Mar-84  
*  
*      a component of lsup.c  
*  
*      version 1.00 as of 25-Mar-84  
*  
*      Copyright 1984 (c) Anthony Skjellum.  
*      All rights reserved.  
*  
*      This program may be freely distributed for all non-commercial  
*      purposes, but may not be sold.  
*  
*      This is a header/definition file which must be included  
*      in any module which utilizes long pointers.  
*  
*****  
  
/*  
   compiler feature toggles:  
   comment out any which don't apply to the compiler in use.  
*/  
  
#define MSUBST           /* macro substitution supported */  
  
/* typedefs */  
  
typedef struct __lword  
{  
    unsigned _addr;          /* address */  
    unsigned _segm;          /* segment */  
} LWORD;  
  
typedef union __lptr  
{  
    long    _llong;          /* long format (for assignments) */  
    char    _lstr[4];          /* character format */  
    LWORD   _lword;          /* long-word format */  
} LPTR;  
  
/* constants */  
  
/* macros */  
  
/* lassign(destination,source): effect assignment of type LPTR */  
  
#ifdef MSUBST  
#define lassign(d,s)    d._llong = s._llong;  
#endif  
  
/* function specifications: */  
  
char lchr();
```

End Listing Three*(Listing Four begins on page 92)*

CP/M® Software

- A>**DBPACK**: Data base management; indexing, sort/search, tabulation, address labels ...
- A>**DBPACK-II**: Advanced data management; payroll, inventory, large & complex data bases ...
- A>**COMCOM**: Communication program; powerful, yet easy to use.
- A>**CPMCPM**: Transfer files (any type) between CP/M computers.
- A>**FILER**: Compresses, archives, catalogs & organizes files.
- A>**UNERA**: Recovers erased files.
- A>**MULTED**: Multi-file text editor.

CP/M is a registered trademark of Digital Research

Configured for a wide variety of systems.
Disk formats include 8-inch, Osborne, Xerox ...

Call or
write
for
information
COMPU-DRAW
1227 Goler House
Rochester, NY 14620
Phone: (716)-454-3188
Dealer
inquiries
invited

MasterCard, Visa & American Express cards welcome.
Separately ordered documentation may be returned
for full refund within 10 days!

It's the writing on the wall

Circle no. 10 on reader service card.

BDS C

The fastest CP/M-80 C compiler available today

Version 1.5 contains some nifty improvements:

The unscrambled, comprehensive new User's Guide comes complete with tutorials, hints, error message explanations and an index.

The CDB symbolic debugger is a valuable new tool, written in C and included in source form. Debug with it, and learn from it.

Hard disk users: You can finally organize your file directories sensibly. During compilation, take advantage of the new path searching ability for all compiler/linker system files. And at run-time, the enhanced file I/O mechanism recognizes user numbers as part of simple filenames, so you can manipulate files located anywhere on your system.

BDS C's powerful original features include dynamic overlays, full library and run-time package source code (to allow customized run-time environments, such as for execution in ROM), plenty of both utilitarian and recreational sample programs, and speed. BDS C takes less time to compile and link programs than any other C compiler around. And the execution speed of that compiled code is typically lightning fast, as the Sieve of Eratosthenes benchmark illustrates. (See the January 1983 BYTE, pg. 303).

BD Software
P.O. Box 9
Brighton, MA 02135
(617) 782-0836

8" SSSD format, \$150
Free shipping on pre-paid orders
Call or write for availability on
other disk formats

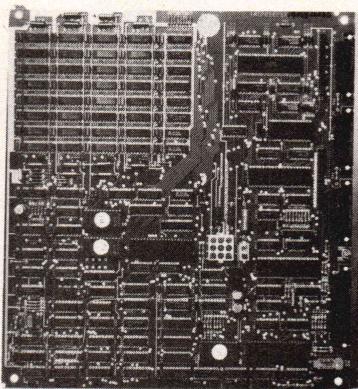
Circle no. 5 on reader service card.

Z80* SINGLE BOARD COMPUTER! 64K RAM — 80 x 24 VIDEO DISPLAY — FLOPPY DISK CONTROLLER RUNS CP/M* 2.2!

BRAND NEW
PC BOARDS
WITH SCHEMATICS!

BOARD MEASURES
11½" x 12½"

ALL ORDERS WILL BE
PROCESSED ON A STRICT,
FIRST COME, FIRST SERVED
BASIS! ORDER EARLY!



\$29.95

(BLANK BOARD WITH
DATA AND ROM'S.)

**NEW
PRICE**

**GROUP SPECIAL:
BUY 6 FOR \$165!**

**USES EASY
TO GET PARTS!**

UNBELIEVABLE LOW PRICE!!! GIANT COMPUTER MANUFACTURER'S SURPLUS!

Recently Xerox Corp. changed designs on their popular 820* computer. These prime, new, 820-1 PC boards were declared as surplus and sold. Their loss is your gain! These boards are 4 layers for lower noise, are solder masked, and have a silk screened component legend. They are absolutely some of the best quality PC boards we have seen, and all have passed final vendor QC. Please note, however, these surplus boards were sold by Xerox to us on an AS IS basis and they will not warranty nor support this part.

We provide complete schematics, ROM'S, and parts lists. If you are an EXPERIENCED computer hacker, this board is for you! Remember, these are prime, unused PC boards! But since we have no control over the quality of parts used to populate the blank board, we must sell these boards as is, without warranty. You will have to do any debugging, if necessary, yourself!

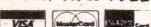
*CP/M TM OF DIGITAL RESEARCH INC. (CALIF.) 820 TM OF XEROX CORP. Z80 TM OF ZILOG

WE ALSO CARRY LS, Z-80, EPROM'S, ETC. SEND FOR FREE CATALOG!

ADD \$2 PER PC BOARD FOR SHIPPING. (USA and Canada)

B. G. MICRO

P. O. Box 280298 Dallas, Texas 75228
(214) 271-5546



TERMS: Orders over \$50 add 85¢ insurance. No COD. Tex. Res. Add 6% Sales Tax. Subject to prior sale. Foreign orders: US funds only. We cannot ship to Mexico. Foreign countries other than Canada add \$6 per board shipping.

Circle no. 6 on reader service card.

Listing Four

```
11sup.asm
a component of lsup.c
Copyright 1984 (c) A. Skjellum. All rights reserved.
version of 25-Mar-84
This routine makes no assumptions about the behavior of the
C compiler in use.
all procedures are "near"

dseg    segment para public 'data'
dseg    ends

cseg    segment para public 'code'
assume cs:cseg,ds:dseg

linc:      increment a long pointer by 1 byte
expects:   es:bx with long pointer to increment
returns:   pointer incremented.
consumes:  es, bx, f, ax

linc      public linc
linc      proc  near
        inc   bx           ; increment low part of word
        or    bx,bx        ; is it zero now?
        jnz  linc_exit    ; no, we are done
        mov   ax,es
        add   ax,1000h      ; another 64k of paragraphs
        mov   es,ax         ; store back to es
linc_exit:
        ret
linc      endp

ldec:      decrement a long pointer by 1 byte
expects:   es:bx with long pointer to decrement
returns:   pointer decremented.
consumes:  es, bx, f, ax

ldec      public ldec
ldec      proc  near
        or    bx,bx        ; zero currently ?
        dec   bx           ; decrement it
        jnz  ldec_exit    ; just decrement low end and exit...
        mov   ax,es
        sub   ax,1000h      ; remove 64k of paragraphs
        mov   es,ax         ; store back to es
ldec_exit:
        ret
ldec      endp
```

```

; ladd:          add a constant to a long pointer

; expects:      es:bx with long pointer's original value
;                ax with unsigned constant to be added
; returns:       pointer with constant added
; consumes:      es, bx, f, ax

;           public ladd
ladd    proc  near
        add   bx,ax           ; add in offset
        jnc  ladd_exit        ; no carry, so we are done.
        mov  ax,es
        add  ax,1000h          ; add 64k of paragraphs
        mov  es,ax             ; and store back to es
ladd_exit:
        ret
ladd    endp

; lsub:          subtract a constant from a long pointer

; expects:      es:bx with long pointer's original value
;                ax with unsigned constant to be subtracted
; returns:       pointer with constant subtracted
; consumes:      es, bx, f, ax

;           public lsub
lsub    proc  near
        sub   bx,ax           ; subtract offset
        jnb  lsub_exit        ; no borrow, so we are done.
        mov  ax,es
        sub  ax,1000h          ; subtract 64k of paragraphs
        mov  es,ax             ; and store back to es
lsub_exit:
        ret
lsub    endp

; lsum:          add a signed offset to a long pointer

; expects:      es:bx with long pointer
;                ax with signed offset
; returns:       pointer with constant added (signed)
; consumes:      es, bx, f, ax

;           public lsum
lsum    proc  near
        or    ax,ax           ; negative?
        jm   lsum_neg
        call ladd             ; do addition
        ret
lsum_neg:
        and  ax,07ffff         ; and out sign flag
        jnz  lsum_neg_ok
        mov  ax,8000h          ; -32768 value (don't treat as 0)
lsum_neg_ok:
        call lsub
        ret
lsum    endp

; lcopy:         copy from one long pointer to another,

```

(Continued on next page)

Listing Four

```
; up to 1024k bytes of data

; expects:    ds:si with src address
;             es:di with dest address
;             ds:cx with length (dx is high order, cx is low order)

; returns:    block copied
;             ds, es intact
; consumes:   ax, cx, f

; this routine uses a copy downward method, to produce
; correct copying for overlapping regions

lcopy    public lcopy
lcopy    proc  near
;
; convert dx into segment form:
;
push    dx      ; save original form of dx
push    cx      ; save low order of long count
and     dx,15  ; smallest meaningful value
xchg   dh,d1  ; switch upper and lower parts
mov    cl,4   ; effect is shift left by 12 bits
shl    dh,cl
pop    cx      ; and recover low order of long count
;
mov    ax,es
add    ax,dx
mov    es,ax
mov    ax,ds
add    ax,dx
mov    ds,ax      ; gross adjustment of segments
pop    dx      ; recover original form of dx
;
add    di,cx      ; adjust dest. ptr to end of area
jnc    no_dest_adj
mov    ax,es
add    ax,1000h  ; add offset
mov    es,ax      ; and store back to segment register
no_dest_adj:
;
; do same work for source pointer:
;
add    si,cx      ; do the addition
jnc    no_mor_adj  ; no more adjustment needed if no carry
mov    ax,ds
add    ax,1000h  ; do the adjustment
mov    ds,ax      ; and store back to ds
;
; at this stage:
;
; es:di is at the last byte of the dest. area
; ds:si is at the last byte of the src. area
;
no_mor_adj:
        std      ; set direction flag for moves
lc_loop:
        or     si,si      ; is si zero ?
```

(Continued on page 96)



THE FULL-FEATURED KEYBOARD EXPANDER for all 8080-8085-8086 computers using CP/M 2.2

MagiKey™ will redefine your keys as character strings ... and transform single keystrokes into commands for programs, words for word processors, data for data bases, and messages for modems. Without hardware or system modifications.

MagiKey™ has more advanced features than any other CP/M keyboard expander. For example:

★ Redefine a key to send the string:

"Run WordStar and edit form letter number 17"

Hit the key, YOU will see the string, but "WS FRMLTR17" will be sent to CP/M. MagiKey™'s CONSOLE REDIRECTION can display messages which are invisible to programs and CP/M ... very useful for recalling key assignments, custom operator prompts, and making CP/M friendlier.

★ Redefine a key to run several programs in sequence. Each program can wait for keyboard input or receive pre-defined commands and data. MagiKey™'s built-in BATCH PROCESSING doesn't use CP/M's SUBMIT, and handles programs that CP/M's XSUB can't.

★ Redefine a key to display the prompt:

"Execute SuperCalc using the spreadsheet file!"

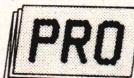
Press the key to display the prompt. Type the file name, hit RETURN, and you're into SuperCalc. When done, a single keystroke saves your updated spreadsheet. You don't have to remember or retype its name. MagiKey™'s RECURSIVE key redefinition mode automatically does it for you.

WE INVITE COMPARISON

\$100

8" SSD, most 5 1/4" formats
add 6% tax in CA
check, VISA, M/C

CP/M (tm) Digital Research
SuperCalc (tm) Sorcim
WordStar (tm) Micropro



microSystems

16609 Sagewood Lane
Poway, California 92064
(619) 693-1022

Circle no. 50 on reader service card.

At Last! bds C . . . Ver.1.5

Including a new dynamic debugger
Still the choice of professionals

- Compiler option to generate special symbol table for new dynamic debugger by David Kirkland. (With the debugger, the distribution package now requires two disks.)
- Takes full advantage of CP/M® 2.x, including random-record read, seek relative to file end, user number prefixes, and better error reporting.

V 1.5 . . . \$120.00

V 1.46 . . . \$115.00

(needs only 1.4 CP/M)

Other C compilers and
C related products
available . . . Call!

TERMS: CHECK,
MONEY ORDER, C.O.D.,
CHARGE CARD

HOURS: 9 am—5 pm
Monday—Friday
(316) 431-0018

IT'S HERE!

MONEY MATH

- Uses BCD internal representation.
- You choose from two types of rounding.
- Configurable exception handling
- Distributed with 12 digits precision. Easily configured for more or less
- Excess 64 exponents

SOURCE INCLUDED \$50.00



Dedicated Micro Systems, Inc.

P.O. Box 481, Chanute, Kansas 66720

include \$2.50 for postage and handling

Circle no. 20 on reader service card.

Six Times Faster!

Super Fast Z80 Assembly Language Development Package

Z80ASM

- Complete Zilog Mnemonic set
- Full Macro facility
- Plain English error messages
- One or two pass operation
- Over 6000 lines/minute
- Supports nested INCLUDE files
- Allows external bytes, words, and expressions (EXT1 * EXT2)
- Labels significant to 16 characters even on externals (SLR Format Only)
- Integral cross-reference
- Upper/lower case optionally significant

- Conditional assembly
- Assemble code for execution at another address (PHASE & DEPHASE)
- Generates COM, HEX, or REL files
- COM files may start at other than 100H
- REL files may be in Microsoft format or SLR format
- Separate PROG, DATA & COMMON address spaces
- Accepts symbol definitions from the console
- Flexible listing facility includes TIME and DATE in listing (CP/M Plus Only)

SLRNK

- Links any combination of SLR format and Microsoft format REL files
- One or two pass operation allows output files up to 64K
- Generates HEX or COM files
- User may specify PROG, DATA, and COMMON loading addresses
- COM may start at other than 100H
- HEX files do not fill empty address space.
- Generate inter-module cross-reference and load map
- Save symbol table to disk in REL format for use in overlay generation
- Declare entry points from console
- The FASTEST Microsoft Compatible Linker available

SPEED!
SPEED!
SPEED!

- Complete Package Includes: Z80ASM, SLRNK, SLRIB - Librarian and Manual for just \$199.99. Manual only, \$30.
- Most formats available for Z80 CP/M, CDOS, & TURBODOS
- Terms: add \$3 shipping US, others \$7. PA add 6% sales tax

For more information or to order, call:

1-800-833-3061

In PA, (412) 282-0864

Or write: SLR SYSTEMS

1622 North Main Street, Butler, Pennsylvania 16001

SLR Systems

Circle no. 66 on reader service card.

Listing Four

```
lodsb          ; get byte ds:[si], decrement si
jnz  no_ds_adj ; no need to adjust if non-zero at start
mov  ax,ds
sub  ax,1000h
mov  ds,ax      ; adjust pointer for next load
no_ds_adj:
    or   di,di      ; is di zero ?
    stosb          ; set byte es:[di] = al, decrement di
    jnz  no_es_adj ; no need to adjust if non-zero at start
    mov  ax,es
    sub  ax,1000h
    mov  es,ax      ; adjust pointer for next store
no_es_adj:
    loop lc_loop   ; copy whole block (--cx, jnz lc_loop)
    dec  dx
    or   dx,dx
    jnz  lc_loop   ; loop over dx counts too
;
; we are done
;
inc  si
inc  di          ; restore to original calling values
ret
lcopy  endp
cseg  ends
end
```

End Listing Four

Listing Five

```
; llint.asm
; version of 25-Mar-84
; a component of lsup.c
; Copyright 1984 (c) A. Skjellum. All rights reserved.
; these routines are setup for Aztec C86 v 1.05i
; all procedures are "near"
;
dseg  segment para public 'data'
dseg  ends
cseg  segment para public 'code'
assume cs:cseg,ds:dseg,es:dseg,ss:dseg
;
; Routines which do not merit calls to portable routines in llsup.asm
```

```

; ds = flptr(lptr,sptr)
; LPTR *lptr;
; char *sptr;
;
; form a long pointer from a "normal" ds relative short pointer (sptr)
; and store at lptr
;
; note: no portable segment (flptr) in llsup.asm since this
; is such a trivial routine.
;
; return value is also ds, should this prove useful
;
        public flptr_
flptr_  proc    near
        mov     bx,sp          ; prepare for argument load
        mov     ax,4[bx]        ; get short pointer ds:ax
        mov     bx,2[bx]        ; get address where to store long pointer
        mov     [bx],ax          ; store low order
        mov     ax,ds
        mov     2[bx],ax        ; store high order
        ret
flptr_  endp          ; return value is also ds

;
; the following four routines are examples of what can
; be done to supplement general routines with specific
; (more efficient ones). Many more variations are
; possible than the two presented here. They follow
; directly from this basic idea:
;
;
; char lchr(lptr)
; LPTR *lptr;
;
; return character pointed to by lptr
;
        public lchr_
lchr_   proc    near
        mov     bx,sp          ; prepare for argument load
        push   ds              ; save ds register
        mov     bx,2[bx]        ; get address of lptr
        mov     ax,[bx]
        mov     ds,2[bx]        ; begin forming pointer
        mov     bx,ax
        sub    ax,ax            ; ds:bx now is valid pointer
        mov     al,[bx]          ; zero whole acc.
        mov     al,[bx]          ; get the character
        pop    ds
        ret
lchr_   endp          ; exit with char in ax

;
; int lint(lptr)
; LPTR *lptr;
;
; return integer or unsigned pointed to by lptr
;
        public lint_
lint_   proc    near
        mov     bx,sp          ; prepare for argument load
        push   ds              ; save ds register
        mov     bx,2[bx]        ; get address of lptr
        mov     ax,[bx]

```

(Continued on next page)

Listing Five

```
        mov     ds,2[bx]          ; begin forming pointer
        mov     bx,ax              ; ds:bx now is valid pointer
        mov     ax,[bx]            ; get the integer or unsigned
        pop    ds                 ; recover old ds value
        ret                 ; and exit with char in ax
lint_  endp

;
; l_stchr(lptr,chr)
; LPTR *lptr;
; char chr;
;
; store character chr at address lptr
;
        public l_stchr_
l_stchr_ proc  near
        mov     bx,sp              ; prepare for argument load
        mov     cl,4[bx]            ; get character
        mov     bx,2[bx]            ; prepare for load of long pointer
        push   ds                 ; save ds segment register
        mov     ax,[bx]            ; begin forming pointer
        mov     ds,2[bx]            ; ds:bx now is valid pointer
        mov     bx,ax              ; store byte
        mov     [bx],cl
        pop    ds
        ret
l_stchr_ endp

;
; l_stint(lptr,val)
; LPTR *lptr;
; int val;
;
; store integer or unsigned at address lptr
;
        public l_stint_
l_stint_ proc  near
        mov     bx,sp              ; prepare for argument load
        mov     cx,4[bx]            ; get integer to store
        mov     bx,2[bx]            ; prepare to form ds:bx with correct
                                   ; storage address
        push   ds                 ; save current ds
        mov     ax,[bx]            ; begin forming pointer
        mov     ds,2[bx]            ; ds:bx now is valid pointer
        mov     bx,ax              ; store the integer
        mov     [bx],cx
        pop    ds
        ret                 ; exit
l_stint_ endp

;
; lload(dest,lptr,len)
; char *dest;
; LPTR *lptr;
; unsigned len;
;
; general purpose copy routine from long data storage to ds: relative
; storage
```

(Continued on page 100)

SUBVERSIVE SOFTWARE

So cheap and useful
it's...dangerous!



SUBVERSIVE SOFTWARE

TPL

The Text Processing Language. A text-file runoff program consisting of a set of text-processing primitive commands from which more complex commands (macros) can be built (as in Logo). Features include:

- Complete customization of text processing through macro definition and expansion, looping structures, and conditional statements;
- Adapts to any printer;
- Pagination;
- Text justification and centering;
- Indexing and tables of contents;
- Superscripts and subscripts;
- Bolding and underlining;
- Multiple headers and footers;
- End notes and footnotes;
- Widow and orphan suppression;
- Floating tables and 'keeps.'

\$50

SUBVERSIVE SOFTWARE

CHROME

Chromatography data analysis program:

- Graphic display of analog data;
- Panning and zooming;
- Automatic peak-finding and baseline calculation;
- Full interactive peak editing;
- Computation of peak areas;
- Strip charts on C. Itoh and EPSON printers.

\$100

SUBVERSIVE SOFTWARE

DBX

Blocked Keyed Data Access Module. Maintains disk files of keyed data. Can be used for bibliographies, glossaries, multikey data base construction, and many other applications.

- Variable-length keys;
- Variable-length data;
- Sequential access and rapid keyed access;
- Single disk access per operation (store, find, delete) in most cases;
- Multiple files;
- Dynamic memory allocation for RAM-resident index and current "page" of entries;
- Includes demonstration program and testbed program.

\$50

SUBVERSIVE SOFTWARE

PLANE

Planimetry program:

- Bit-pad entry of cross sections;
- Real-time turtlegraphics display;
- Calculation of areas;
- Saves calculations to text file.

\$100

SUBVERSIVE SOFTWARE

PDMS

The Pascal Data Management System. A user-oriented data management system in which numeric and alphanumeric data are stored in tables with named columns and numbered rows. Currently being used for dozens of different kinds of business and scientific applications, from inventory management to laboratory data analysis. Includes over 20 Pascal programs; more than 10,000 lines of code. Main features include:

- Maximum of 32,767 rows per file;
- Maximum of 400 characters per row, and 40 columns per table;
- Full-screen editing of rows and columns, with scrolling, windowing, global search/replace, and other editing features;
- Sorting, copying, merging, and reducing routines;
- Mailing label program;
- Reporting program generates reports with control breaks, totals and subtotals, and selects rows by field value; many other reporting features;
- Cross-tabulation, correlations, and multiple regression;
- Video-display-handling module;
- Disk-file-handling module.

Many other features. UCSD formats only.

\$250

SUBVERSIVE SOFTWARE

ZED

Full-screen text editor; designed to be used either with TPL or by itself.

- Full cursor control;
- Insert mode with word wrap;
- 'Paint' mode;
- Single-keystroke or dual-keystroke commands;
- Command synonyms;
- Global search and replace;
- Block move, block copy, and block delete.

\$50

SUBVERSIVE SOFTWARE

SCINTILLA

A log logit curve fitting program for radio-immunologic data; must be used with PDMS (described above).

- Multiple protocol files;
- Quality control files;
- Four-parameter non-linear curve fit.

UCSD formats only.

\$250

SUBVERSIVE SOFTWARE

MINT

A terminal emulation program for communication between computers of any size.

- User-configurable uploading and downloading of files;
- X-ON/X-OFF and EOB/ACK protocols;
- Interrupt-driven serial input (for Prometheus Versacard in Apple II);
- Printer-logging.

\$50

For more information, call 919-942-1411. To order, use form below or call our toll-free number: 1-800-XPASCAL

Check appropriate boxes:

FORMAT

- 8" UCSD SSSD
- 5 1/4" Apple Pascal
- 5 1/4" UCSD IBM PC 320k
- 8" CP/M SSSD
- 5 1/4" IBM MS-DOS
- 5 1/4" CP/M Osborne

PRODUCT

- DBX
- PDMS
- TPL
- ZED
- MINT
- SCINTILLA
- CHROME
- PLANE

PRICE

- \$50
- \$250
- \$50
- \$50
- \$50
- \$250
- \$100
- \$100

Name

Address

MasterCard

(Please include card # and expiration date)

VISA

Check

C.O.D.

SUBVERSIVE SOFTWARE

A division of Pascal & Associates,

135 East Rosemary St., Chapel Hill, NC 27514

Listing Five

```
; we assume es = ds for Aztec C explicitly here
; due to convenient 8086 instructions, the portable function would
; consist merely of a "cld", "rep movsb" sequence followed by a return.
; Therefore, no portable lload is included in llsup.asm

; lload_
lload_  public lload_
        proc  near
        mov   bx,sp      ; prepare for argument load
        push  ds          ; save Aztec ds segment
        push  si
        push  di          ; save source and destination indices
        ;
        mov   cx,6[bx]   ; get length for move
        mov   di,2[bx]   ; es:di now has the destination address
        mov   bx,4[bx]   ; prepare to load long ptr
        mov   si,[bx]    ; get low order
        mov   ds,2[bx]   ; and then high order
        cld
        rep   movsb      ; do the move
        pop   di
        pop   si
        pop   ds
        ret
lload_  endp

; 1stor(lptr,src,len)
; LPTR *lptr;
; char *src;
; unsigned len;
;
; Reverse of lload: this routine copies data from ds:src to lptr
; Once again, there is no llsup analog.

; 1stor_
1stor_  public 1stor_
        proc  near
        mov   bx,sp      ; prepare for argument load
        push  es
        push  di
        push  si          ; save registers as required by Aztec C.
        mov   cx,6[bx]   ; get length of move
        mov   si,4[bx]   ; ds:si now contains source index
        mov   bx,2[bx]   ; prepare to form es:di
        mov   di,[bx]
        mov   es,2[bx]
        rep   movsb      ; move the data
        pop   si
        pop   di
        pop   es          ; restore registers and exit
1stor_  endp

; -----
;
; routines which call portable subroutines in llsup.asm
```

```

; -----
; linc(lptr)
; LPTR *lptr;

; increment a long pointer by 1

        public linc_
linc_  proc    near
        extrn linc:near
        mov    bx,sp           ; prepare for argument load
        push   es              ; save es value from caller
        mov    bx,2[bx]
        push   bx              ; address where answer will go
        mov    es,2[bx]          ; get segment
        mov    bx,[bx]           ; and address
        call   linc             ; do the work
        pop    ax
        xchg   ax,bx
        mov    [bx],ax
        mov    2[bx],es          ; store the value
        pop    es              ; recover the old value
        ret    es              ; and exit
linc_  endp

; -----
; ldec(lptr)
; LPTR *lptr;

; decrement a long pointer by 1

        public ldec_
ldec_  proc    near
        extrn ldec:near
        mov    bx,sp           ; prepare for argument load
        push   es              ; preserve es
        mov    bx,2[bx]
        push   bx              ; get address ds:bx
        mov    ax,[bx]           ; address where answer will go
        mov    es,2[bx]
        mov    bx,ax
        call   ldec             ; do the decrement
        pop    ax
        xchg   ax,bx
        mov    [bx],ax
        mov    2[bx],es          ; store the value
        pop    es              ; recover it for sake of caller
        ret    es              ; and then exit
ldec_  endp

; -----
; ladd(lptr,offset)
; LPTR *lptr;
; unsigned offset;

; add unsigned offset to a long pointer

        public ladd_
ladd_  proc    near
        extrn ladd:near
        mov    bx,sp           ; prepare for argument load
        push   es              ; save es value of caller
        mov    ax,4[bx]          ; get the offset to ax

```

(Continued on next page)

Listing Five

```
        mov      bx,2[bx]
        push    bx          ; address where answer will go too.
        mov      cx,[bx]
        mov      es,2[bx]
        mov      bx,cx
        call    ladd         ; do the addition
        pop     ax
        xchg   ax,bx
        mov     [bx],ax
        mov     2[bx],es      ; store the value
        pop     es          ; recover old es value
        ret      ; and then exit
ladd_  endp

;
; lsub(lptr,offset)
; LPTR *lptr;
; unsigned offset;
;
; subtract unsigned offset from a long pointer
;
        public lsub_
lsub_  proc  near
        extrn lsub:near
        mov      bx,sp        ; prepare for argument load
        push    es          ; preserve es
        mov      ax,4[bx]     ; get the offset
        mov      bx,2[bx]
        push    bx          ; store answer at this addr. too.
        mov      cx,[bx]
        mov      es,2[bx]
        mov      bx,cx
        call    lsub         ; do the subtraction
        pop     ax
        xchg   ax,bx
        mov     [bx],ax
        mov     2[bx],es      ; store the value
        pop     es          ; restore es
        ret      ; and then exit
lsub_  endp

;
; lsum(lptr,offset)
; LPTR *lptr;
; int offset;
;
; add signed offset to a long pointer
;
        public lsum_
lsum_  proc  near
        extrn lsum:near
        mov      bx,sp        ; prepare for argument load
        push    es          ; preserve caller's es
        mov      ax,4[bx]     ; get the signed offset
        mov      bx,2[bx]
        push    bx
        mov      cx,[bx]
```

```

    mov    es,2[bx]
    mov    bx,cs
    call   lsum           ; do the signed addition
    pop    ax
    xchg  ax,bx
    mov    [bx],ax
    mov    2[bx],es        ; store the value
    pop    es
    ret    ; exit.

lsum_  endp

```

```

; lcopy(dest,src,len)
; LPTR *dest;
; LPTR *src;
; long len;      (treated as a long unsigned quantity)
;
; copy from src to dest, len bytes.
;
; note this routine can be used to copy arbitrarily large chunks of memory
;
public lcopy_
lcopy_ proc near
extrn lcopy:near
    mov    bx,sp           ; prepare for argument load
    push   ds

```

(Continued on next page)



WRITE

The Writer's Really Incredible Text Editor lives up to its name! It's designed for creative and report writing and carefully protects your text. Includes many features missing from WordStar, such as sorted directory listings, fast scrolling, and trial printing to the screen. All editing commands are single-letter and easily changed. Detailed manual included. WRITE is \$239.00.

WORKMAN & ASSOCIATES

112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401

All US orders are postpaid. We ship from stock on many formats, including: 8", Apple, Osborne, KayPro, Otrona, Epson, Morrow, Lobo, Zenith, Xerox. Please request our new catalog. We welcome COD orders.

Circle no. 77 on reader service card.

GREP in C

The UNIX* regular expression recognizer

MAIN

Wildcard expansion & pipes for Aztec C

All programs come with complete source code, in C. Price: \$35 each; \$50 together.

For more information or complete catalogue:

SOFTWARE ENGINEERING CONSULTANTS

P. O. BOX 5679
BERKELEY, CA 94705

(415) 548-6268

* A trademark of Bell Laboratories

Circle no. 69 on reader service card.

Listing Five

```
push    es          ; save segment registers
push    di
push    si          ; save these registers for Aztec C
;
mov    cx,6[bx]    ; get length (low order)
mov    dx,8[bx]    ; high order of length
mov    ax,2[bx]    ; get ds:ax as pointer to dest.
xchg   ax,bx
mov    di,[bx]
mov    es,2[bx]
mov    bx,ax
mov    bx,4[bx]    ; get ds:bx as pointer for dest.
mov    si,[bx]
mov    ds,2[bx]    ; get long pointer
;
call    lcopy
;
pop    si
pop    di
pop    es
pop    ds
ret
lcopy_ endp
cseg    ends
end
```

End Listing Five

Listing Six

```
/*
env.c                               created: 25-Mar-84

This package echoes the environment to the standard output.

example of using long pointers with lsup package.

This is set-up to work with Aztec C.

by Anthony Skjellum. (C) 1984. All rights reserved.
Released for non-commercial purposes only.

This program echoes the environment block to the console.
In effect, this is the same as the DOS 2.0 SET command.
Nevertheless, it illustrates the usefulness of long pointers.
```

The following changes were made to the \$begin
routine of the Aztec C 1.05i module calldos.asm:

i) a new global variable called envseg was created

```
envseg_ segment word common 'data'
$envdat dw 0
$envseg dw ?
envseg_ ends
```

(Continued on page 106)

Dr. Dobb's Journal NOW AT BIGGER SAVINGS!

\$35.40
\$25.00

If you take advantage of this special offer you save over \$10 off newsstand prices, that's a 30% savings!

Please charge my: Visa MasterCard American Express
 Payment enclosed Bill me later

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____ 3092

Offer good in USA only. Foreign rates upon request. Please allow up to six weeks for first issue. This offer good until August 31, 1984.

A publication of M&T Publishing.

Dr. Dobb's Journal

For the Experienced in Microcomputing

Reader Service Card

To obtain information about products or services mentioned in this issue, circle the appropriate number listed below. Use bottom row to vote for best article in issue. This card is valid for 90 days from issue date. Use only one card per person.

June 1984, No. 92

Name _____

Address _____

City/State/Zip _____ Phone (____) _____

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81
82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108
109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135

Articles: 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214

Comments:

Purchase of Magazine:

Subscription
 Computer Store
 Newsstand
 Bookstore
 Passed on by friend/colleague
 Other _____

Dr. Dobb's Journal

For the Experienced in Microcomputing

Reader Service Card

To obtain information about products or services mentioned in this issue, circle the appropriate number listed below. Use bottom row to vote for best article in issue. This card is valid for 90 days from issue date. Use only one card per person.

June 1984, No. 92

Name _____

Address _____

City/State/Zip _____ Phone (____) _____

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81
82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108
109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135

Articles: 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214

Comments:

Purchase of Magazine:

Subscription
 Computer Store
 Newsstand
 Bookstore
 Passed on by friend/colleague
 Other _____



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 756 MENLO PARK, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal

For the Experienced in Microcomputing

P. O. Box 27809
San Diego, CA 92128



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 756 MENLO PARK, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal

For the Experienced in Microcomputing

2464 EMBARCADERO WAY
PALO ALTO, CA 94303



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 756 MENLO PARK, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal

For the Experienced in Microcomputing

2464 EMBARCADERO WAY
PALO ALTO, CA 94303

It's a headache, of sorts.

Whether your mailing list, accounting package, DBMS, or other end-user application is currently under development or already on the market, you've probably discovered that external sorting can be a difficult, costly, and time-consuming headache. Especially in today's floppy-disk storage environment.

That's why we developed BEAMSORT™, the revolutionary *in-place* OEM sorting module. The older algorithms eat up valuable space on your user's often over-crowded diskettes. By installing BEAMSORT™ you get today's technology at a fraction of its true development cost, and you've got one less problem to worry about.



*CPM, MS-DOS, dBase II, and SuperSort are trademarks of Digital Research, Microsoft, Ashton-Tate, and Micropro, respectively.

BEAMSORT™ runs under CPM-80*, CPM-86*, MS-DOS*, and PC-DOS, supports multiple-volume files, ASCII, Microsoft .RAN, and dBASE II*.DBF file formats, and interfaces to all major languages. Custom interfacing, operating environments, and file formats are available.

What about performance? BEAMSORT™ runs SuperSort™'s own benchmark faster than SuperSort™ does! It's amazing, but don't take our word for it. Write us on your company letterhead for our OEM evaluation kit. You must see this for yourself!

For fast relief.



Phlexible Data Systems, Inc.

3017 Bateman Street, Berkeley, CA 94705
(415) 540-7181

Circle no. 46 on reader service card.

GET "C" APPLICATIONS OFF TO A FLYING START WITH

C-TREE™

RECORD MANAGEMENT SUBSYSTEM

- Advanced B+ Tree Structure
- Fast And Efficient
- Unlimited # Of Keys
- Keys May Be Duplicate, LIFO/ FIFO, Modifiable
- Record Locking Calls
- Sequential Access
- Utilities To Add/Delete Keys And Fields, Rebuild Files
- Error Processing Interface
- Store Data Dictionary In File

C-SORT™

SORT/SELECT/MERGE SUBSYSTEM

- Advanced Quick/Tournament Sort
- Sort B-Tree or Sequential Files
- Automatically Uses All Available Memory
- Sort On Any Number/Type Of Field
- Select Records According To User-Specified Criteria
- Creates Tag (Index) Sorting File
- Automatic Interface To B-Tree

ordering information

SINGLE UNIT LICENSE

\$99 plus shipping

Format: 5 1/4" Disk, MS-DOS-compatible. Linkable 8086-file format modules for Lattice-C Compilers, others soon. Complete documentation.

SOURCE CODE OPTION

\$149 plus shipping.

"C" Source Code is also available; requires license.

MULTIPLE COPY OPTION

Multiple copies of object code may be made with this license at a very low unit cost.

Telephone Orders Accepted
Visa/Mastercard
(512) 476-8356

AccuData Software
Dept. D-6
P.O. Box 6502
Austin, Texas 78762

Circle no. 1 on reader service card.

DeSmet C

The fastest 8088 C Compiler available

FULL DEVELOPMENT PACKAGE

- C Compiler
- Assembler
- Linker and Librarian
- Full-Screen Editor
- Newsletter for bugs/updates

SYMBOLIC DEBUGGER

- Monitor and change variables by name using C expressions
- Multi-Screen support for debugging PC graphics and interactive systems
- Optionally display C source during execution
- Breakpoint by Function and Line #

COMPLETE IMPLEMENTATION

- Both 1.0 and 2.0 DOS support
- Everything in K&R (incl. STDIO)
- Intel assembler mnemonics
- Both 8087 and Software Floating Point

OUTSTANDING PERFORMANCE

Sieve Benchmark

COMPILE	4 Sec. RAM —
	22 Sec FDISK
LINK	6 Sec. RAM —
	34 Sec. FDISK
RUN	12 Sec.
SIZE	8192 bytes

DeSmet C
Development Package \$159

To Order Specify:

Machine _____

OS MS-DOS CP/M-86
Disk 8" 5 1/4 SS 5 1/4 DS

CWARE
CORPORATION

P.O. BOX 710097
San Jose, CA 95171-0097
(408) 736-6905

California residents add sales tax. Shipping: U.S. no charge, Canada add \$5, elsewhere add \$15. Checks must be on a U.S. Bank and in US Dollars.

Circle no. 17 on reader service card.

Listing Six

ii) On entry to \$begin, when es contains the program segment prefix (PSF), es:[2ch] contains the segment address of the environment. This segment address is stored into the second word of envseg_ (ie \$envseg).

The environment may now be referred to through the external LPTR envseg.

iii) If DOS 2.0 allocation is to be used, be sure to shrink the program size using the SETBLOCK function. This must also be done in \$begin where the psp, ds, segments are both available.

*/

```
#include <stdio.h>
#include "lsup.h"      /* support for long pointers */

extern LPTR envseg;    /* envseg is a structure of type LPTR */

main(argc,argv)
int argc;
char *argv[];
{
    char chr;
    int i;
    LPTR lptr;

    lassign(lptr,envseg); /* get long pointer to environment */

    while(1)      /* loop */
    {
        chr = lchr(&lptr);      /* get the next byte */

        if(!chr) /* we have hit the end of the environment */
            break;

        while((chr = lchr(&lptr))) /* get characters of string */
        {
            putchar(chr);      /* write them to console */
            linc(&lptr);      /* increment pointer */
        }

        linc(&lptr);      /* pass the zero byte just encountered */

        putchar('\n');    /* add new line between entries */
    } /* end while(1) */
}
```

3

End Listings

OF INTEREST

by Michael Wiesenber

Track Zero

I knew that this year's **West Coast Computer Faire** would be hoppin' from the moment I tried to get in on Saturday. "No problem," *DDJ* editor **Renny Wiggins** had said on Thursday. "Your pass is waiting for you. I saw it myself."

It seems that *DDJ* passes can be filed under four possible names: **Doctor Dobb's Journal**, **People's Computer Company**, **M & T Publishing, Inc.**, and **Business Software** (the new sister publication). I checked all with no success.

Officials kept shunting me from one line to another, from one person with the authority to grant tickets to another. They were less than impressed with my "business card" that Renny had assured me would be my entree if all else failed: the name of the magazine, the new address, but not my name.

In desperation I called the *DDJ* booth. New editor-in-chief **Mike Swaine** came out to assist. After 15 minutes of further fruitless official shuttling and line waiting, I asked Mike, "What would **Usasi** do?" "Sneak in the side door."

Finally **Renny** showed up. He found my ticket misplaced among the press passes, and I was in.

The show itself was a bore. There was nothing to match last year's **Perfect Software** laser and fog spaceship, nor the numerous **robots** walking among the throngs.

This year the most exciting offerings were several speech synthesizers that faithfully reproduced verbally anything passers-by typed on their keyboards. Apple had a 16-foot **Mac** mockup, but we've seen giant screens before, particularly those with wavering, fuzzy images.

Probably the best show in town was the **Friends of Dr. Dobb's Party** that night at the **Vorpal Gallery**.

The **Vorpal Gallery**, filled with world renowned **modern art**, including many **Escher** originals, was a classy place for **M & T** to announce its licensing agreement with **Dr. Dobb's Journal**.

The food was excellent — piping hot meatballs in sauce, vegetables with several dips, stuffed mushrooms, puff pastries, brie — and the many varieties of wine never stopped flowing.

I asked **Timothy Leary** if he was

going to get into psychedelic software. He said that it was already here, that computers were becoming the drugs of the eighties.

I shook hands with **Robin Williams**, who was carrying a plastic shopping bag full of computer goodies and literature just like the rest of us.

Sat Tara Singh Khalsa, president of **Kriya Systems**, and devoted to the idea of **programmer as superstar**, held court in turban and white Sikh outfit.

Art Kleiner, editor of the **Whole Earth Software Catalog and Review**, was there, as were **Tony Bove** and **Cheryl Rhodes**, copublishers of **The Users' Guide**.

Dick Heiser, who started the first computer store ever, showed up, as did **Gordon Eubanks**, developer of **CBASIC**.

I spoke with **John Draper**, better known in his persona of the infamous **Cap'n Crunch**, phone phreak and publicizer of various rainbow-colored boxes. **Gerald Schmidt**, the undercover FBI informant who has been called "the most knowledgeable person on computer crime in the country," met Crunch face-to-face for the first time at the party.

Roger Gregory, systems anarchist, and **Ted Nelson**, both of **Project Xanadu**, floated about the canapés.

John Markoff, West Coast editor for **Byte**, and **Paul Freiberger**, who has the same position with **Popular Computing**, talked with their former associate, **David Needle**, news editor of **InfoWorld**.

Carl Helmers, the first editor of **Byte**, was seen, as were **David Ahl**, publisher of **Creative Computing** and **Sync**, and **Elizabeth Staples**, editor of **Creative Computing**.

Robert Harp, president of **Corona Systems**, wandered around.

Chris Terry, technical editor of **Microsystems**, was in evidence.

Midway through the party, **John Barry**, managing editor of **InfoWorld Books**, and **Jack Klyster**, mad inventor of the **klystron tube**, came in together.

While punching it down **Market Street** in my **Volvo** station wagon, listening to **Antonin Dvorak's New World Symphony** on the stereo, I said to myself, "I hope they give me real business cards before next year's Faire."

Punching Diskettes

The **Disk Notcher** from **Quorum** makes another write enable notch so you can use the flip side of diskettes. For **Apple II+** and **IIe**, **Certifix** formats the new side, examines the surface, locks out flaws so they can't be used, displays and saves status report, and initializes with **DOS 3.3**. Both products cost \$29.95, with 64 write protect tabs and 32 diskette labels thrown in, or **Certifix** alone for \$24.95, and the **Disk Notcher** tool alone for \$15. Add \$1.50 p. and h., and **Californians** add sales tax. **Reader Service No. 101**.

A Likely Case

Anvil Cases fit most microcomputers and are virtually indestructible. They are foam-lined with aluminum exterior. Some can be customized by users for unusual configurations. In addition to the portable cases, rack-mount cases are available for heavier equipment. **Anvil** has been making these cases for 20 years for the music industry, to take sound systems on the road. **Reader Service No. 103**.

Dazzle Your Friends

One of the best graphics displays I saw at the Faire was produced by the **Graphics Dazzler** from **Sigma Designs**, a plug-in for the **IBM PC**, with 1024 x 1024 x 4 maximum display memory, 640 x 200 standard viewable display, 640 x 400 in "interlaced" mode. You get two display planes for four colors standard, or four display planes for 16 with the **Graphics Enhancer** stack. The board is compatible with all color and black-and-white monitors used with the PC. The display area can be panned over the 1024 x 1024 display memory, with a 1-to-16 zoom over any display area. All the work is performed by **NEC 7220** display controller software, which also has hardware line drawing, area filling, and arc drawing. There is a light pen interface, and you get graphic display software and **DOS**.

boot program. The Dazzler is \$895, Enhancer \$695. The Maximizer, at \$395, has, in one card, 64K RAM, expandable to 384K, parallel interface for printer or bidirectional I/O, RS-232C serial interface, clock/calendar with battery backup, RAM disk, printer spooler, and, optionally, second RS-232C (\$60) and a game control adapter that supports four paddles or two joysticks (\$40). Sigma also offers various expansion cards, video cards, and hard disk subsystems that add from 10 to 33 Mb from \$1695 to \$4295. **Reader Service No. 105.**

Two More Processors for Heath/Zenith

The H-1000, from Technical Micro Systems, is a Z80/8086 plug-in replacement upgrade for H89/Z89 that needs no modifications. The Z80 runs at 2/4 MHz, and the 8086 at 8 MHz. The board comes with 256K RAM, expandable to 1 Mb. It has five I/O slots, and is fully compatible with all Heath/Zenith peripherals. It runs all Heath/Zenith software without modification, and is compatible with Zenith Z100 and IBM PC. You get your choice of MS-DOS or CP/M-86. TMSI seems a bit confused about its prices. The specifications brochure claims 256K RAM standard, while the price list cites the basic board with 128K for \$995, and upgrade to 256K for another \$250. Expansion to 512K is \$1695, and to 1 Mb, \$3495. They will also sell it to you as a complete system in a modified H89/Z89 case: 128K and one 100K disk drive, \$2495; 256K and one 320K drive, \$2995; or 256K and two 320K drives, \$3295. TMSI also has a lot of software for the board and computer, including Concurrent CP/M-86, MP/M-86, RAM disk utilities, accounting and communications packages, data base, languages legal, planning, spreadsheet, word processing. You can also discuss their making a custom configuration for you. **Reader Service No. 107.**

Mass Marketing Software

Paul Terrell believes that software should be vended at K-Marts, Tower Records, Long's, and 7-Elevens on reprogrammable video game cartridges. The president of Romox was showing off his products at the Faire, including programs for as little as \$7.95. When

you get tired of one, just bring back the cartridge, pay the clerk from \$7.95 to \$15, and get a new one. Not just games, but education, business, etc., are available now for Atari computers and 2600, Commodore 64, Vic 20, TI 99/44A, and coming soon for IBM PC, PCjr, Coleco and MSX. I saw the ROM programmer at work. You can see a list of all games and get a short preview of any game on the list. **Reader Service No. 109.**

Get GUMMed

The Gurus of Unix Meeting of Minds (GUMM) takes place Wednesday, April 1, 2076 (check that in your perpetual calendar program), 14 feet above the ground directly in front of the Milpitas Gumps. Members will grep each other by the hand (after intro), yacc a lot, smoke filtered chroots in pipes, chown with forks, use the wc (unless uuclean), fseek nice zombie processes, strip, and sleep, but not, we hope, od. Three days will be devoted to discussion of the ramifications of whodo. Two seconds have been allotted for a complete rundown of all the user-friendly features of Unix. Seminars include "Everything You Know is Wrong," led by Tom Kempson, "Batman or Cat:MAN?" led by Richie Dennis, "cc C? Si! Si!" led by Kerwin Bernighan, and "Document Unix? Are You Kidding?" led by Jan Yeats. No **Reader Service No.** is necessary because all GUGUS (Gurus of Unix Group of Users) already know everything we could tell them.

Simply the Best

Computer Literacy Book Shop claims to be the best computer bookstore in the world, and the only one devoted exclusively to computers. They have over 4500 titles, and if you don't find what you want, they'll order it. Owners Dan Doernberg and Rachel Unkefer have acted as consultants to the Whole Earth Software Review. **Reader Service No. 111.**

Pay for It if You Like It

PC-Write, from Quicksort, has the right marketing idea. This excellent word processor for IBM PC and PCjr

(which some reviewers claim is better than the old standbys; it has all the features of the \$400 word processors and a few more) costs \$10 if you get it from the publisher, or it's free if you get it from a friend. Quicksort encourages distribution of copies. They ask only that if you like it you send them \$75. They'll send you a bound copy of their manual, give you telephone support, a free copy of the next revision, Pascal and assembly source, and a \$25 commission when anyone else registers from a copy of your registered diskette. **Reader Service No. 113.**

Contact Points

Anvil Cases, Inc., 4128 Temple City Blvd., Rosemead, CA 91770; (213) 575-8614.

Computer Literacy Book Shop, 520 Lawrence Expressway, Suite 310, Sunnyvale, CA 94086; (408) 730-9955.

Quicksort, 219 First N #224, Seattle, WA 98109; (206) 282-0452.

Quorum International, Unltd., Industrial Park Station, Box 2134, Oakland, CA 94614; (415) 552-8240.

Romox, Inc., 476 Vandell Way, Campbell, CA 95008; (408) 374-7200.

Sigma Designs, Inc., 2990 Scott Blvd., Santa Clara, CA 95050; (408) 496-0536.

Technical Micro Systems, Inc., Dept. H, 366 Cloverdale, Box 7227, Ann Arbor, MI 48107; (313) 994-0784.



Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 200.

MATH SUBROUTINE LIBRARY

Now, a library of **Numerical Methods Subroutines** for use with your **FORTRAN** programs.

Over sixty subroutines of:
FUNCTIONS
INTEGRATION
MATRICES
NON LINEAR SYSTEMS
INTERPOLATION
LINEAR SYSTEMS
POLYNOMIALS
DIFFERENTIAL EQ

Versions available for several FORTRAN compilers running under **CP/M-80** and **MS-DOS (PC-DOS)**.

Cost. \$250.
Manual available. \$25.

microSUB: MATH

CP/M and MS-DOS are trademarks of Digital Research Corp. and Microsoft Corp., respectively.

Circle no. 24 on reader service card.

ADVERTISER INDEX

Reader Service No.	Advertiser	Page No.	Reader Service No.	Advertiser	Page No.	Reader Service No.	Advertiser	Page No.
1	AccuData Software	105	26	GTEK	47	52	Protocols	107
2	Amber Systems, Inc.	53	27	Hallock Systems Consultants	77	53	Puterparts	110
3	Ashton-Tate	3-5	28	Harvard Softworks	42	54	Quelo	47
4	Avocet Systems, Inc.	25	29	Integral Quality	51	55	Quest Research	53
5	BD Software	91	30	JV Software	107	56	Quic-N-Easi Products, Inc.	11
6	B. G. Micro	91	31	Key Solutions	35	57	Quicksoft	67
7	Bonnie Blue Software	17	32	Laboratory Microsystems	78	58	Rational Systems	85
8	Borland International	112	33	Lattice, Inc.	73	* Edward Ream	65	
9	California Digital Engineering	73	34	Leo Electronics, Inc.	67	60	The Redding Group	62
10	Compu-Draw	91	35	Life Boat Associates	59	61	Revasco	79
11	Computer Design Labs	21	36	Manx Software	8	62	S. C. Digital	107
12	Computer Friends	72	37	MicroMethod	27	63	SemiDisk Systems	71
13	Computer Innovations	41	38	MicroMotion	79	64	Shaw Laboratories	49
14	Coriolis Company	107	39	Microprocessors Unlimited	107	65	Simpliway	107
15	Creative Solutions	69	40	MU Software	53	66	SLR Systems	95
16	C User's Group	110	41	Next Generation Systems	63	67	Softcraft	2
17	C Ware	105	42	Novum Organum	47	68	Software Building Blocks	67
78	DDJ Bound Volume	79	43	Occo, Inc.	16	69	Software Engineering Consultants	103
18	Data Access Corporation	111	44	Overbeek Enterprises	65	70	Software Toolworks	65
19	Datalight	61	45	Pascal & Associates	99	71	Southern Computer Company	51
20	Dedicated Microsystems	95	46	Phlexible Data Systems	105	72	Telecon Systems, Inc.	61
21	D & W Digital	31	47	Proa Corporation	35	73	Visual Age	61
22	Ecosoft, Inc.	51	48	ProCode International	35	74	VOCS	27
23	Farware	107	49	The Programmer's Shop	85	75	Mark Williams & Company	43
24	Foehn Consulting	109	50	PRO Microsystems	95	76	Western Wares	107
25	GGM Systems Inc.	85	51	PRO Microsystems	107	77	Workman & Associates	103

Available now.

The Best of Both Worlds

High-performance
CompuPro Hardware
with high-performance
Cromemco Software.

Cromemco's MC68000-Z80 CROMIX multi-tasking
operating system with drivers to support CompuPro hardware.
(requires Cromemco DPU)

Minimum configuration:

DPU - SystemSupport1 - Disk1 - Interfacer3 or 4 - 192K RAM

CROMIX with drivers to support minimum configuration \$890.
Special drivers only \$295.

The following products can be added to any 8 or 8-16 bit CROMIX system:

SCSI hard disk drivers \$195.
15Mb formatted hard disk subsystem \$2095.
30Mb (two drives) \$2995.

MDrive-H drivers \$195.
CompuPro 512K MDrive-H hardware \$1695.

Other drivers in development... custom inquiries welcome.

Authorized CompuPro Dealer.

PuterParts®

2004 4th Avenue

P.O. Box 12339

Seattle, WA 98111-4339

Telephone (206) 682-2590

OPEN
2-6 Tuesday-Saturday

"PuterParts" is a registered trademark

of Katharos Companies

Additional trademarks: Z80, Zilog, MC68000, Motorola, CROMIX, DPU, Cromemco, SystemSupport1, Disk1, Interfacer3, 4, MDrive-H, CompuPro.

Circle no. 53 on reader service card.

*Elegance
Power
Speed*



Users' Group

Supporting All C Users

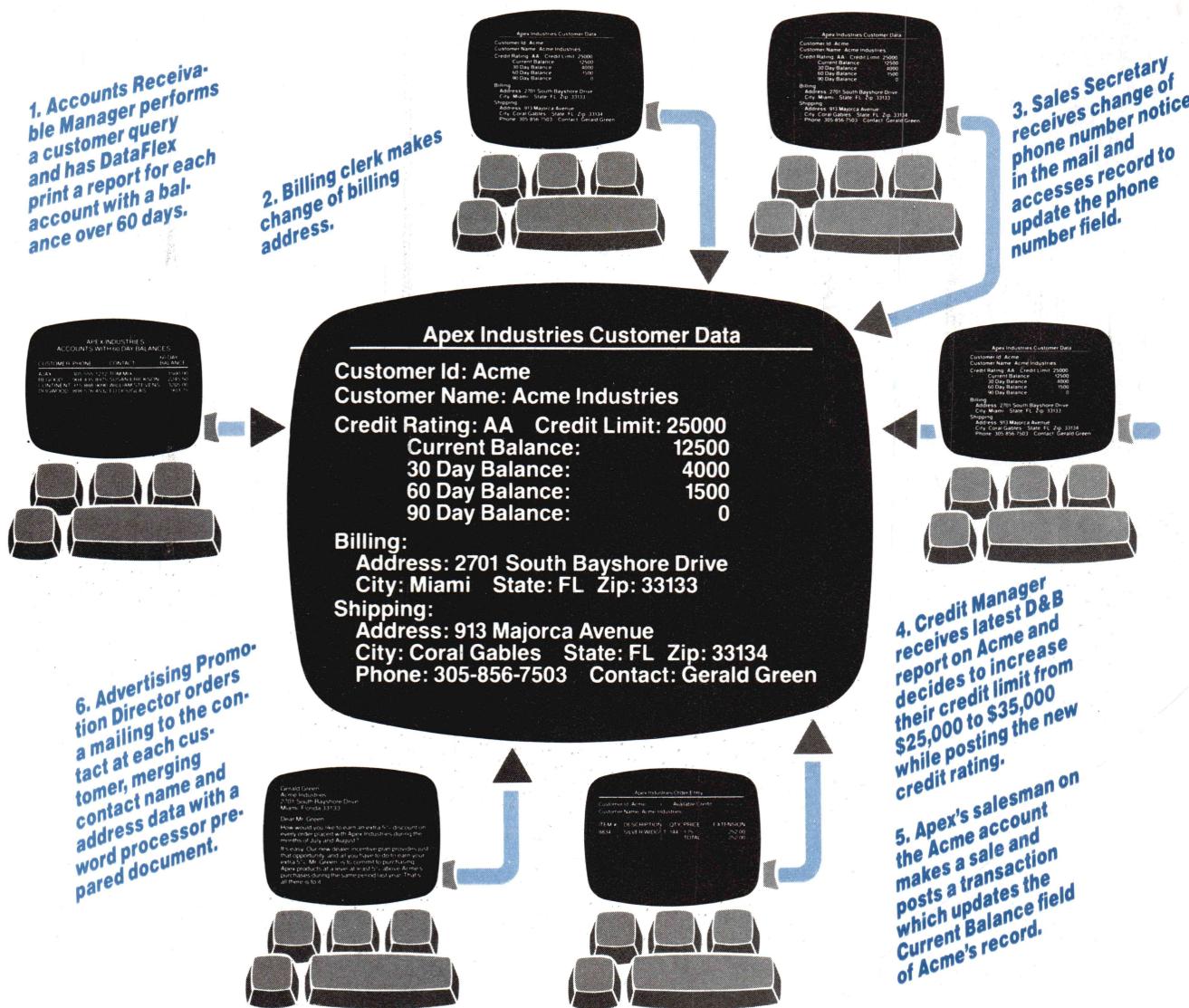
Box 287

Yates Center, KS 66783

Circle no. 16 on reader service card.

ALL AT ONCE!

AND NEVER A "LOCKED OUT" USER!



DataFlex is the only application development database which **automatically** gives you true multi-user capabilities. Other systems can lock you out of records or entire files for the full time they are being used by someone else. DataFlex, however, locks only the data being changed, and **only** during the micro-seconds it

takes to actually write it to the file! The updated record is then immediately available. The number of users who can access, and change, records at the same time is limited only by the number of terminals on your system or network. Call or write today for all the details on DataFlex...the true multi-user database.

DATAFLEX™

DATA ACCESS CORPORATION

8525 SW 129 Terrace, Miami, FL 33156 (305) 238-0012
Telex 469021 DATA ACCESS CI

See us at



Compatible with CP/M-80, MSDOS networks, MP/M-86, Novell Sharenet, PC-Net, DMS Hi-net, TurboDOS multi-user, Molecular N-Star, Televideo MmmOST, Action DPC/OS, IBM PC w/Corvus, OMNINET, 3Com EtherSeries and Micromation M/NET.

MSDOS is a trademark of Microsoft. CP/M and MP/M are trademarks of Digital Research.

May 22-25, 1984
Georgia World Congress Center,
Atlanta, Georgia

Booth #3424

Circle no. 18 on reader service card.

ANNOUNCING . . . VERSION 2.0



"What I think the computer industry is headed for: well documented, standard, plenty of good features, and a reasonable price."

Jerry Pournelle,
Byte, February 1984

"The Perfect Pascal"

Alan R. Miller,
Interface Age, January 1984

If you already own Turbo Pascal version 1.0, you can upgrade to 2.0 for \$29.95. Just send in your old master with your check. (Manual update included of course).

EXTENDED PASCAL FOR YOUR
IBM PC, PC jr., APPLE CP/M,
MSDOS, CP/M 86, CCP/M,
OR CP/M 80

NOW . . .

WITH
WINDOWING
\$49.95

NEW FEATURES

WINDOWING!

... This is a real shocker. On the IBM PC or PC jr. you'll now have a procedure to program windows. . . . Any part of the screen can be selected as a window and all output will automatically go to this part of the screen only. As many windows as you please can be used from the same program.

AUTOMATIC OVERLAYS!

... No addresses or memory space to calculate, you simply specify OVERLAY and TURBO PASCAL will do the rest.

GRAPHICS, SOUND AND COLOR SUPPORT

... For your IBM PC or JR!

FULL HEAP MANAGEMENT!

... via dispose procedure.

OPTIONAL 8087 SUPPORT!

... Available for an additional charge.

If you have a 16 bit computer with the 8087 math chip—your number crunching programs will execute up to 10X faster!

ORDER YOUR COPY OF TURBO PASCAL VERSION 2.0 TODAY

For VISA and MasterCard orders call toll free:

1-800-227-2400 x968

In CA:

1-800-772-2666 x968

(lines open 24 hrs, 7 days a week)

Dealer & Distributor Inquiries welcome

408-438-8400

CHOOSE ONE (please add
\$5.00 for shipping and handling
for U.S. orders)

Turbo Pascal 2.0 \$49.95
 Turbo Pascal 2.0 with
8087 support \$89.95

Update (1.0 to 2.0) Must
be accompanied by the
original master \$29.95

Update (1.0 to 8087) Must
be accompanied by the
original master \$69.95

Check Money Order
VISA MasterCard
Card #:
Exp. date: Shipped UPS

My system is: 8 bit 16 bit
Operating System: CP/M 80
CP/M 86 MS DOS PC DOS
Computer: Disk Format:
Please be sure model number & format are correct.

NAME:

ADDRESS:

CITY/STATE/ZIP:

TELEPHONE:

California residents add 6% sales tax. Outside U.S.A. add \$15.00. (If outside of U.S.A. payment must be by bank draft payable in the U.S. and in U.S. dollars.) Sorry, no C.O.D. or Purchase Orders. B11

 **BORLAND**
INTERNATIONAL

Borland International
4113 Scotts Valley Drive
Scotts Valley, California 95066
TELEX: 172373

Circle no. 8 on reader service card.